Desarrollo de un videojuego de plataformas 2D en Unity

Universitat Politècnica de València
Campus d'Alcoi

Grado en Ingeniería Informática

Trabajo de fin de grado

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Alumno: *Mariola Trenzano Álvarez*
Tutor: *Jordi Joan Linares Pellicer*
Curso Académico: *2016/2017*

ÍNDICE

1. INTRODUCCIÓN

- **Descripción del proyecto:**

El proyecto realizado trata sobre desarrollar un videojuego de plataformas con estilo 2D con el motor de videojuegos Unity.

Este juego ira dirigido a todos los públicos, ya que no contiene ningún tipo de insulto, ninguna escena violenta con sangre, etc y podrán disfrutarlo los Smartphone que tengan como sistema operativo Android.

- **Motivación:**

La mayor motivación que he tenido con este proyecto es que nunca antes me han enseñado a crear videojuegos a excepción de la única asignatura optativa de 4º, por tanto, es un mundo nuevo que descubrir para poder desafiarme a mí misma y para aumentar los conocimientos sobre el desarrollo de videojuegos.

Me motiva el hecho de crear algo desde 0 partiendo de mi creatividad, de mis conocimientos básicos de programación y de tener claro que en un futuro quiero estar trabajando para este sector.

Mi objetivo con este proyecto es demostrar que con interés y tiempo se puede hacer un buen juego que pueda inspirar e atraer a más gente que les apasione tanto como a mí los videojuegos.

- ***Originalidad de la idea:***

  La idea de hacer este trabajo fin de grado fue porque quería complementar mi afición con el trabajo profesional, ya que como he dicho anteriormente, quiero dedicarme profesionalmente a esta industria.

  Además de que me guste la idea, esto me va a servir como portafolio para que el día de mañana si alguna empresa necesite ver que soy capaz de hacer, pueda mostrárselo con hechos y no solo con palabras.

- ***Estado del arte:***

A día de hoy, hacer un videojuego está al alcance de casi todo el mundo, gracias a que existen motores de videojuegos gratuitos para que cualquier persona que esté interesada en este tema, pueda ser autodidacta y realizar sus pruebas para aprender mucho más de lo que pueda saber.

Cuando aparecieron los primeros videojuegos, si alguien quería hacer uno, era realmente mucho más difícil que ahora porque requería de un buen capital, ya que al no haber tantos motores de videojuegos como hay ahora, cada empresa debía hacer el suyo propio y eso implica más dinero y más tiempo.

En este caso, podemos hacer referencia al motor de videojuegos que está arrasando en este momento que se denomina Unreal Engine que la empresa que lo creó fue Epic Games y lo que hizo fue dar el paso de abrirse para que la gente pudiera usarlo gratuitamente. Unity como hablaremos más adelante, es otro motor de videojuegos que se puede usar gratuitamente al igual que este.

Actualmente, hay tantos motores de videojuegos gratuitos que hay demasiados juegos que salen a la venta en un mismo año, esto hace que a los jugadores que quieran probarlos, realmente no tienen tiempo para dedicarles a todos y seleccionarán los juegos que más les llame la atención.

Por así decirlo, hay 3 tipos de empresas que desarrollan videojuegos:

- **Empresas oficiales.**
- **Empresas independientes (Indies).**
- **Personas amateurs.**

En cuanto a empresas oficiales, me refiero a empresas como Microsoft Studios, Sony Interactive Entertainment, Nintendo, etc.

Son empresas que normalmente hacen juegos triple “A”, esto quiere decir que es un juego que tiene mucho dinero para poder invertir, que tiene muchos recursos para poderse hacer y mucho tiempo para poder dedicarle para que salga lo más perfecto posible.

Algunos ejemplos de juegos triple “A” que han realizado serían:

Rise of the Tomb Raider *(lanzamiento 2015)*

Zelda breath of the wild *(lanzamiento 2017)*

Horizon zero dawn *(lanzamiento 2017)*

Mario Kart 8 Deluxe *(lanzamiento 2017)*

Final Fantasy XV *(lanzamiento 2016)*

Uncharted 4 *(lanzamiento 2016)*

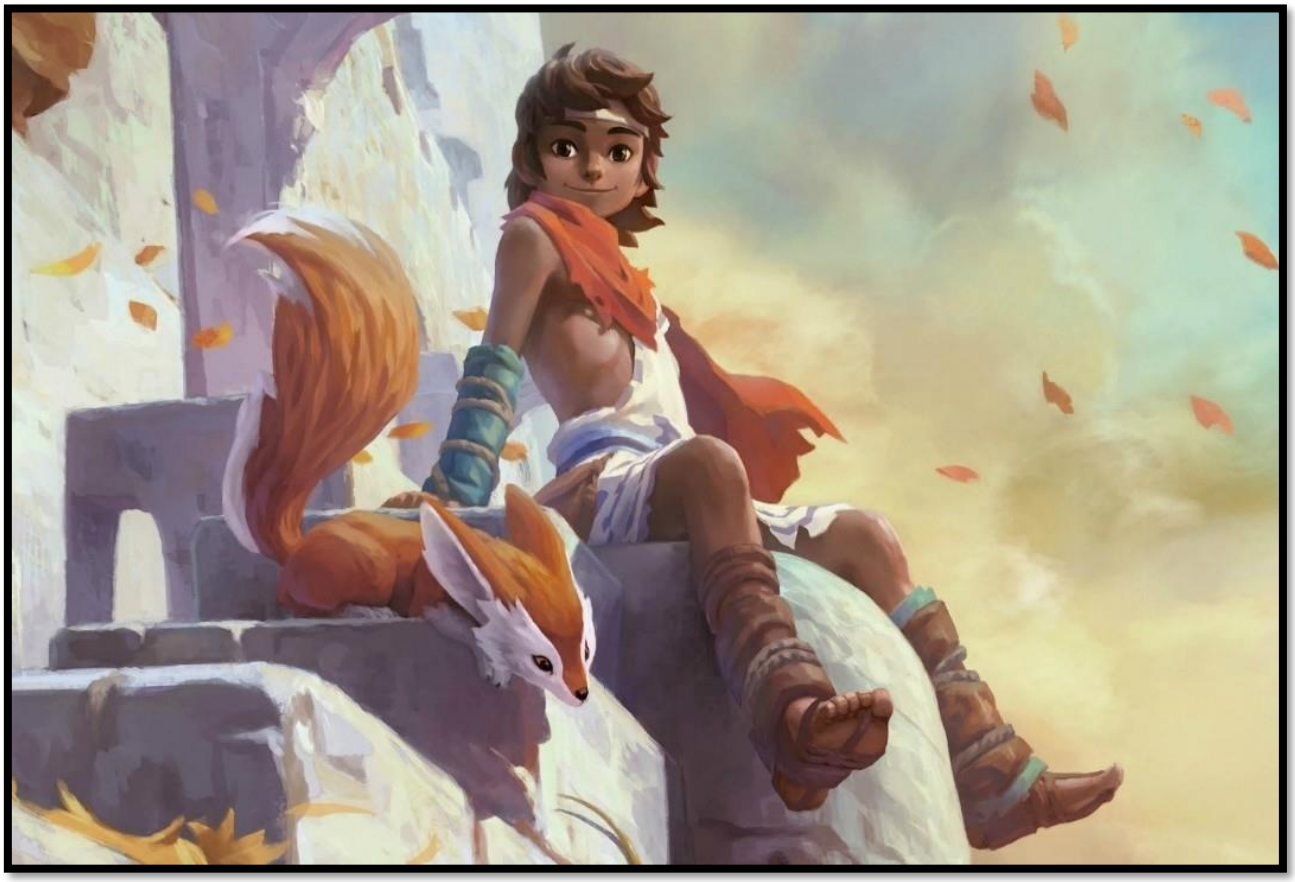
No todas las empresas pueden aportar gran cantidad de dinero, por ejemplo, sería el caso de una empresa independiente o de alguna persona solitaria que crea su propio juego.

Una empresa independiente muchas veces se denomina ***Indie***, es una empresa que no tiene el apoyo de una distribuidora de videojuegos y que los videojuegos que suelen desarrollar son distintos a los productos de las empresas oficiales que siguen las tendencias del momento. Este tipo de empresa suele contar con un equipo de 30 personas más o menos, en cambio una empresa como la que hemos hablado anteriormente suele tener un equipo de cientos de personas.

Ejemplos de empresas independientes serían Polytron, Tequila Works, Thatgamecompany, Plastic, Playdead, etc.

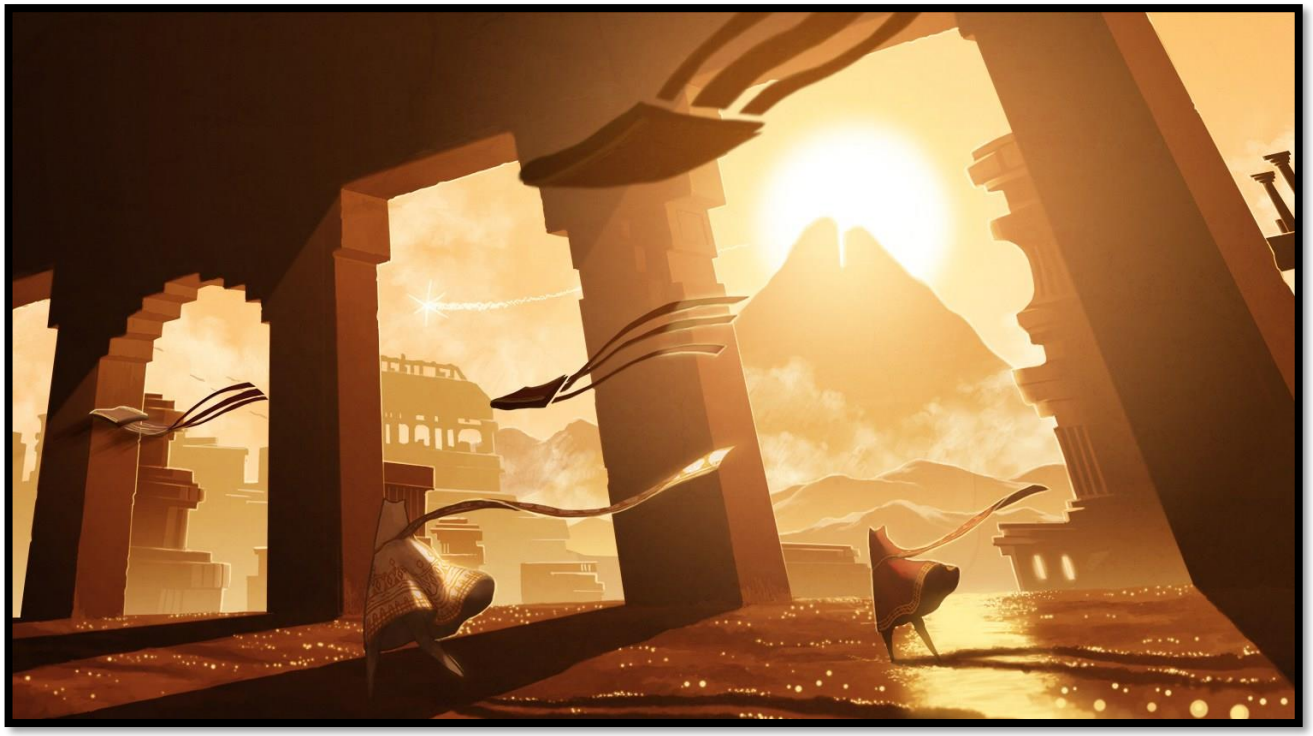
Algunos ejemplos de juegos Indie que han realizado serían:

Fez *(lanzamiento 2012)*

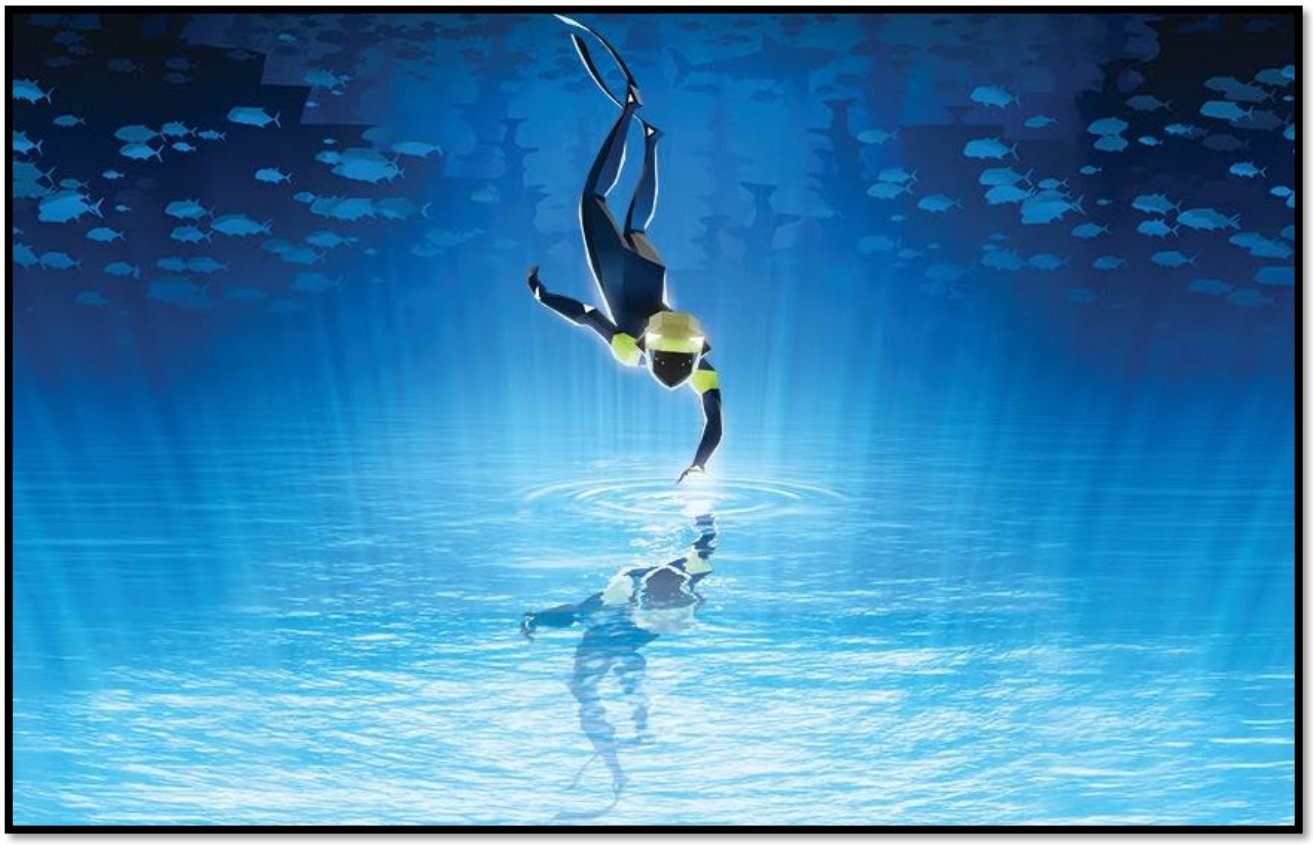


Rime *(lanzamiento 2017)*

Bound *(lanzamiento 2016)*



Journey *(lanzamiento 2015)*



Abzú *(lanzamiento 2016)*

Por último, con personas amateurs me refiero a las personas que han desarrollado su propio juego en su casa, sin la ayuda de un equipo de desarrolladores que trabajan en la industria, si no, alguna persona que haya hecho su propio videojuego.

Algunos ejemplos serían:

Stardew Valley *(lanzamiento 2016)*

Dust: An Elysian Tail *(lanzamiento 2013)*

Banished *(lanzamiento 2014)*

Thomas was alone *(lanzamiento 2012)*

- **Objetivos:**

El objetivo principal de este proyecto, como he dicho anteriormente, es crear un juego en 2D de tipo plataformas para dispositivos móviles, el cual pueda mostrar en un futuro como portafolio para las empresas que quiera entrar a trabajar.

Además, tengo como objetivo aprender a diseñar niveles, porque es un tema que parece que sea sencillo, pero tiene mucha complejidad, porque tienes que medir las cosas con lupa, es decir, si pones por ejemplo una caja en el nivel, esa caja estará ahí por alguna razón, no porque sí, ser un buen diseñador de niveles requiere de mucho análisis y práctica.

A parte de usar el programa Unity, he usado diferentes herramientas que nombraré más adelante, por tanto, también es una oportunidad para aprender a usar mejor o desde cero herramientas que pueden servir para el desarrollo de videojuegos.

- ***Público destinatario:***

El público al que va dirigido es para todas aquellas personas que les guste pasar un rato divertido y entretenido jugando sin tener que preocuparse de si hay insultos, violencia, etc. Va a ser un juego para dar buenas sensaciones al usuario, por tanto, es para todos los públicos.

Además, el entorno gráfico que tiene el juego es como si perteneciera a un mundo de fantasía, por eso pienso que también es un punto a favor para que todo el mundo lo vea factible para cualquier persona y no se defina para un tipo de público en concreto de cierta edad, sino, un juego que pueda jugar cualquier persona que sepa manejar un teléfono móvil y que sea capaz de transmitir ciertas cosas a adultos y otras cosas para los más niños.

2. HERRAMIENTAS DE DESARROLLO

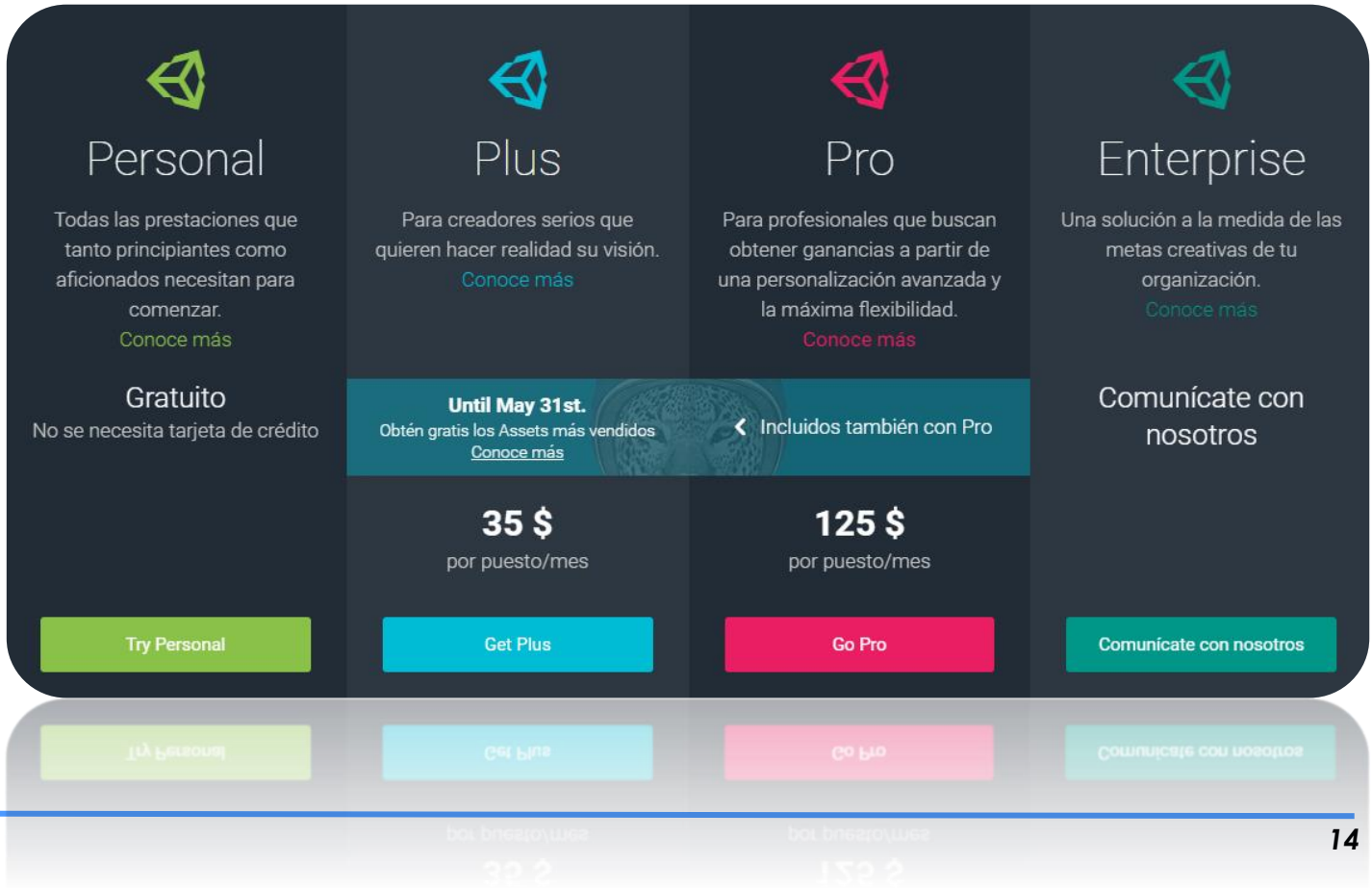
- **Breve introducción a Unity:**

Unity es un motor de videojuegos multiplataforma creado por Unity Technologies que actualmente está siendo muy utilizado por los desarrolladores.

Una de las razones de su uso es porque se pueden desarrollar juegos para diversas plataformas, como, por ejemplo:

- PC: Microsoft Windows, OS X y Linux.
- Móviles: Windows Phone, iOS y Android.
- Consolas: Xbox, Playstation y Nintendo.
- Televisiones.
- Realidad virtual.
- Web: WebGL.

Unity consta de 4 versiones que son las siguientes:



Todas las versiones tienen las mismas características, lo único que cambia cada versión es que dependiendo de la cantidad de monetización que se adquiera, Unity obliga al desarrollador del juego a coger una versión u otra. Por ejemplo, si el desarrollador no llega a 100.000$ puede coger la versión gratuita (Personal), si se pasa de ahí y no llega a los 200.000$, tendrá que coger la versión Plus.

Por último, si el desarrollador tiene beneficios de más de 200.000$ cogería la versión Pro o la Enterprise, las dos no tienen límite de beneficios. La única diferencia es que la Enterprise es a nivel de empresa y la Pro no.

- **Otras herramientas usadas:**

Otras herramientas que me han hecho falta a la hora de desarrollar este proyecto, han sido las siguientes:

- ***Photoshop:***
  Para editar imágenes para la interfaz, para los menús de los niveles, el tipo de letra, etc.

- ***Trello:***
  Herramienta de trabajo para gestionar todas las tareas que he tenido que realizar durante este proyecto y llevar un control de las mismas.
  Entrando a la página web https://trello.com/ puedes registrarte gratuitamente y utilizar su herramienta.

- ***Piskel:***
  Editor online y libre para crear sprites que he usado para el juego con el estilo que se denomina pixel art.

  Herramienta muy útil donde puedes trabajar en el mismo navegador entrando en la página web *http://www.piskelapp.com/* o te lo puedes descargar para utilizarlo en tu ordenador.

  Más adelante, adjunto el GDD donde aparecen todos los sprites que he hecho.

- ***Microsoft Project:***
  Programa para realizar el diagrama de Gantt que se trata más adelante en este documento.

- ***Github:***
  Repositorio donde he podido almacenar todo mi proyecto con todos los cambios efectuados desde el principio hasta el final, mediante commits que se realizan cada vez que hayas hecho cambios importantes en tu proyecto.

- ***Google drive:***
  Es la nube de Google que la he utilizado para poder almacenar todos los documentos, imágenes del proyecto e incluso poder realizar dicha encuesta y obtener resultados que se tratarán más adelante.
  Es una herramienta muy útil, ya que puedes obtener todos tus recursos en cualquier ubicación.

- ***Visual Studio:***
  Es el programa que he utilizado para programar todos los scripts del proyecto.

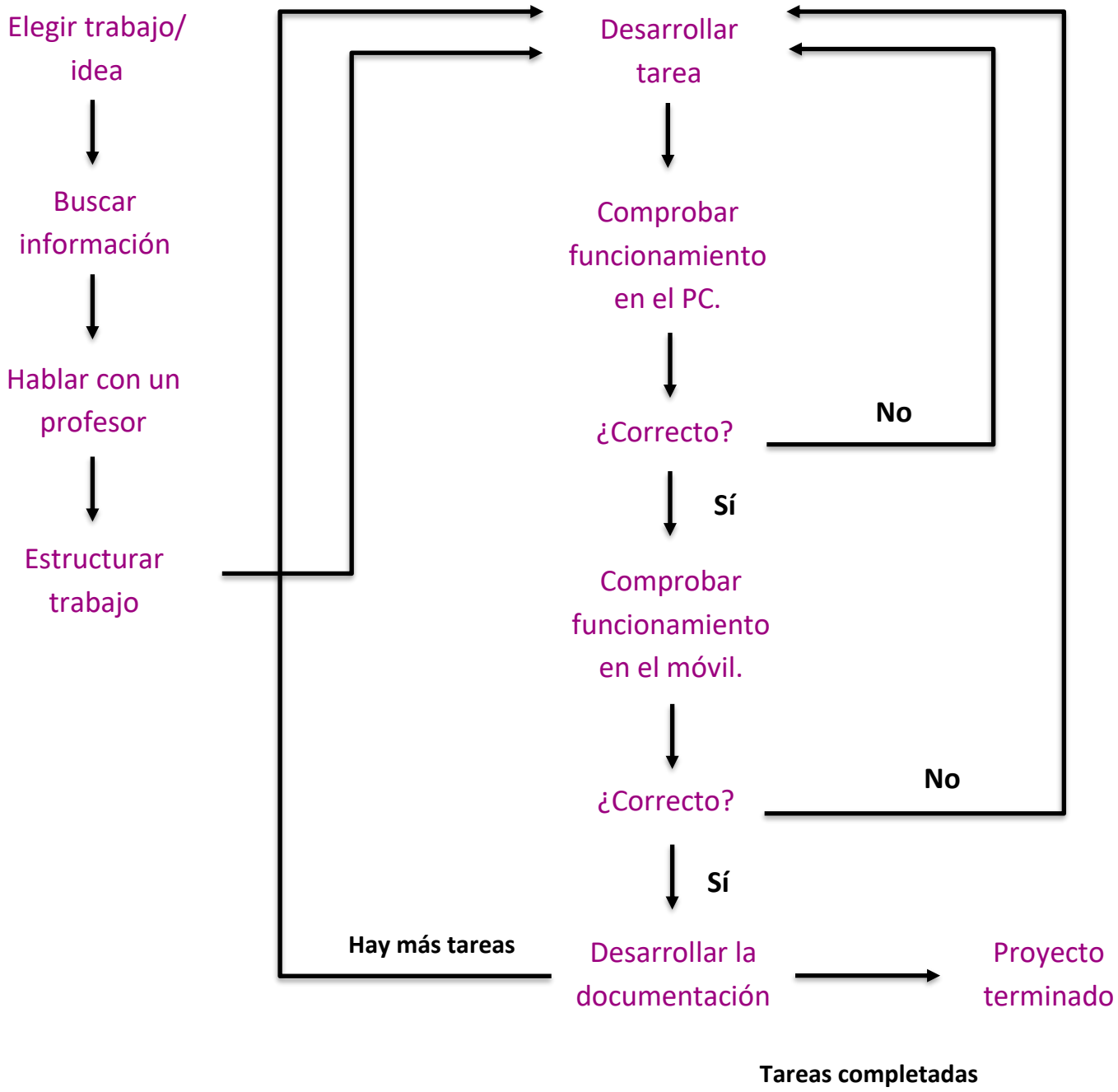
  Unity por defecto tiene el programa de MonoDevelop, pero me ha gustado más utilizar Visual Studio, ya que tiene muchas más configuraciones y detalles que el otro no tiene.

- ***Google Play Console:***
  Plataforma web que te ofrece Google para publicar tus aplicaciones en la tienda de Play Store y poder observar todo lo que tiene que ver con ellas.

3. DISEÑO E IMPLEMENTACIÓN

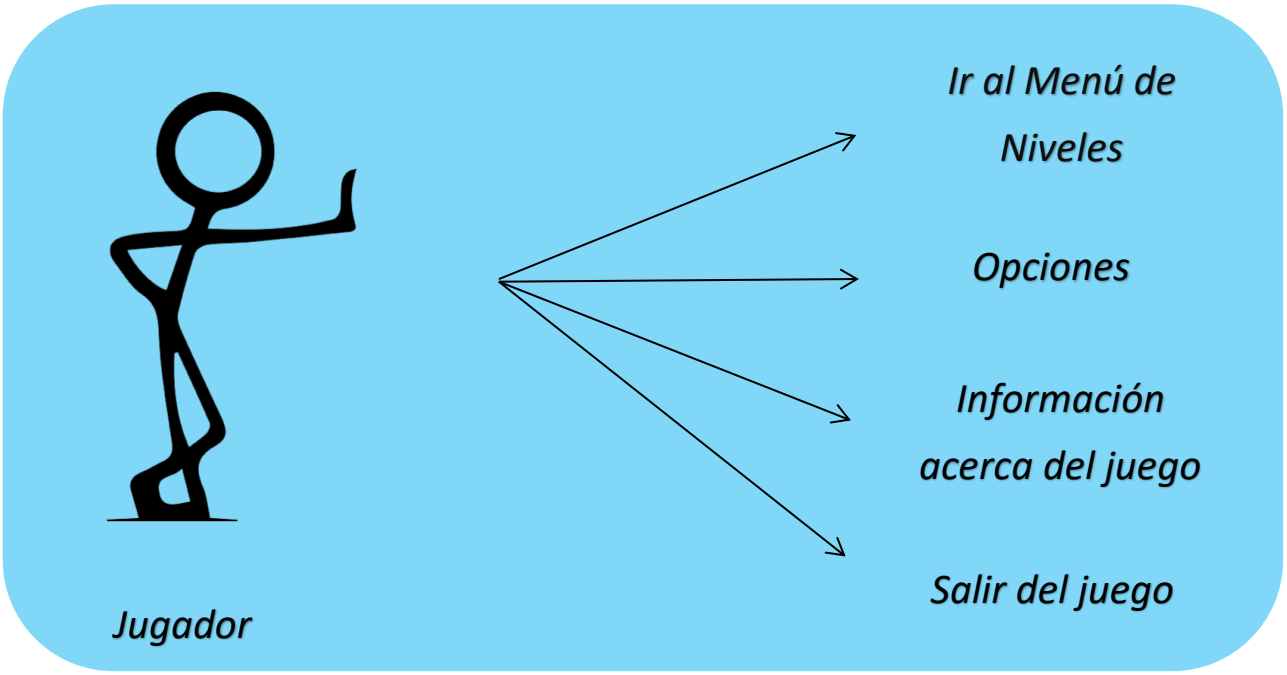
- **Diagramas y casos de uso:**

A continuación, se muestra un diagrama donde define la estructura del proyecto.

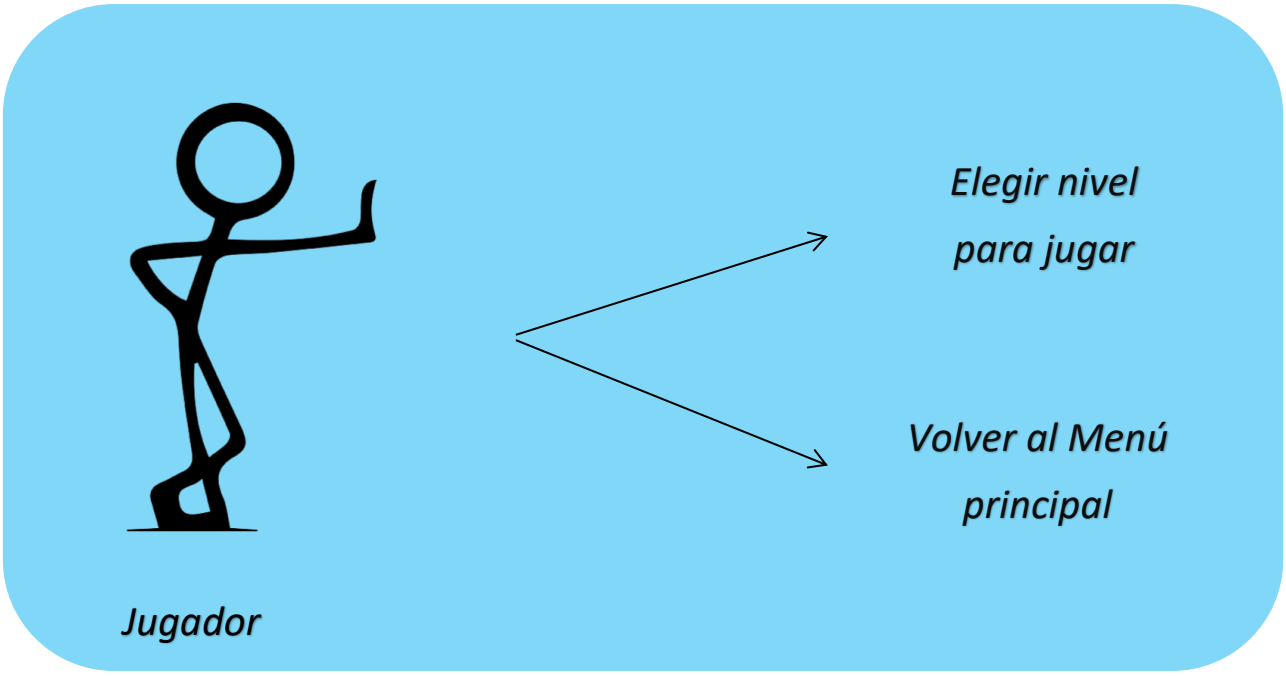


Los casos de uso que se presentan en el juego son los siguientes:

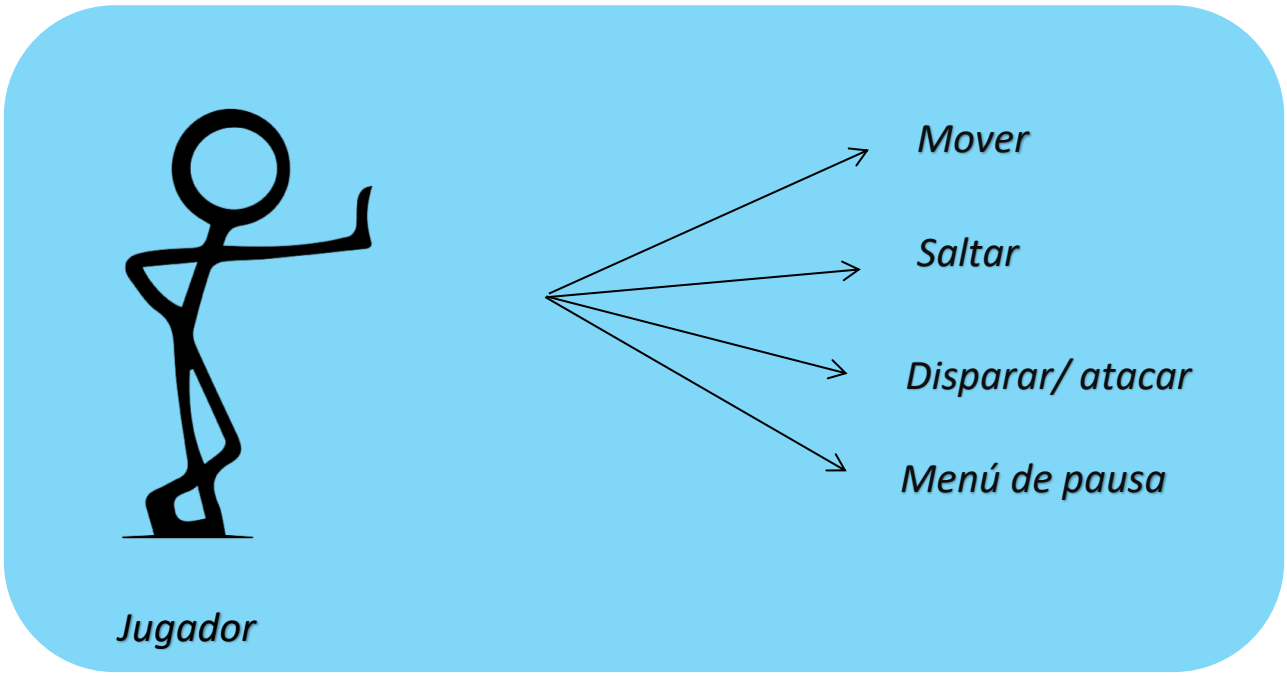
Menú principal



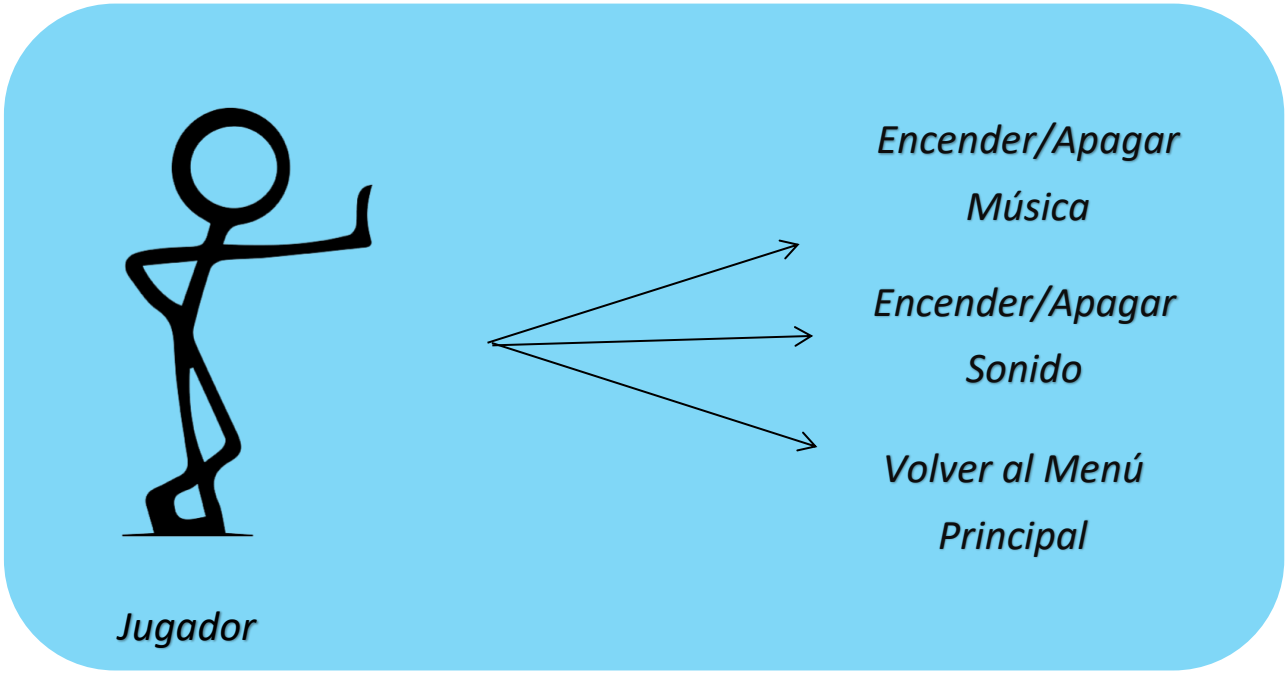
Menú de Niveles



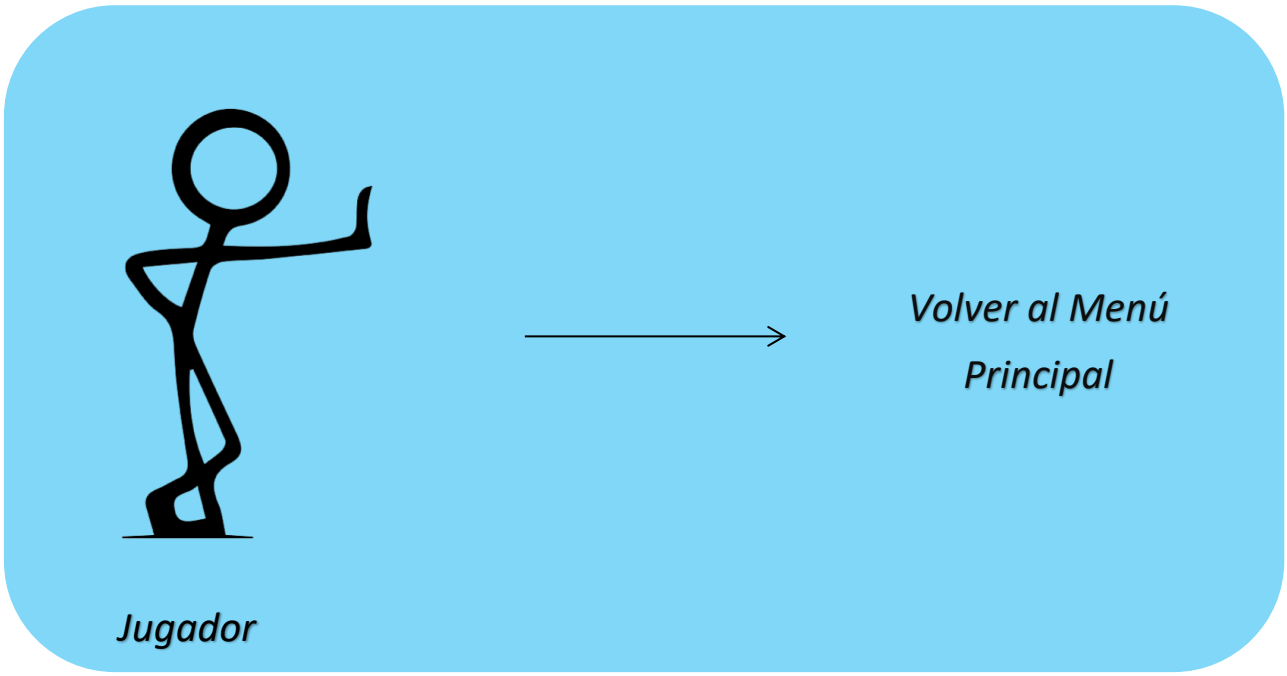
Dentro de un nivel



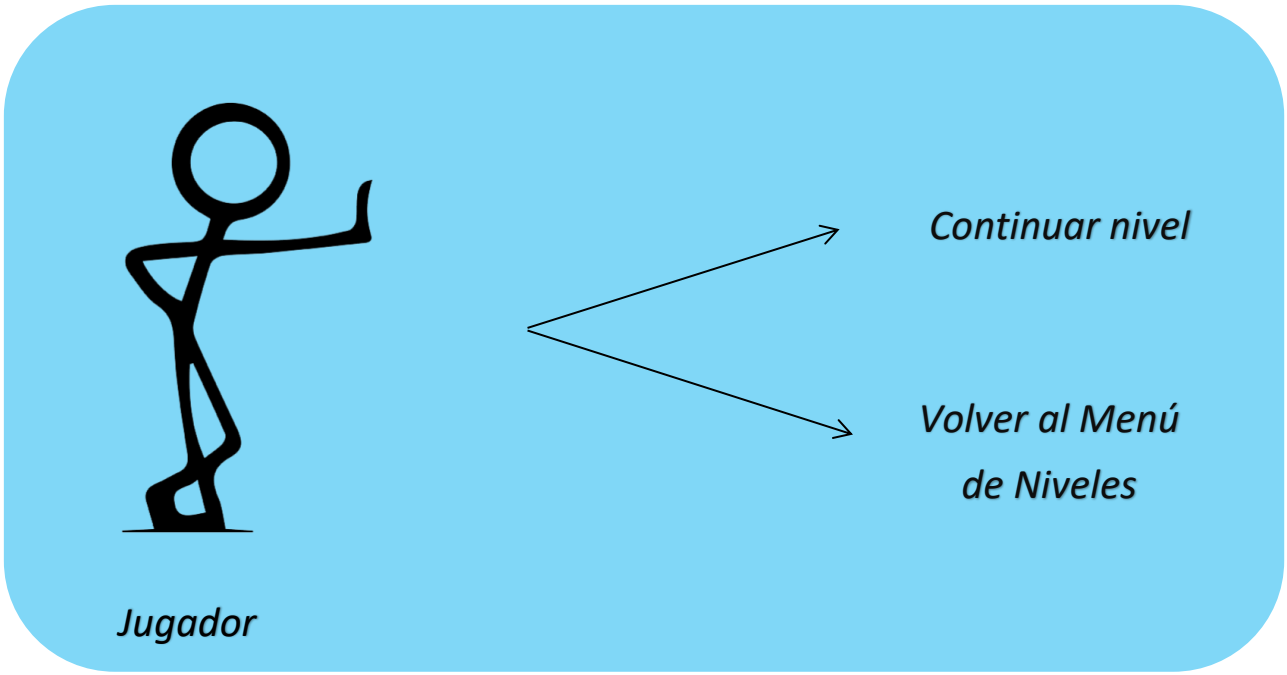
Opciones



Información acerca del juego



Menú de pausa



Nivel completado / Pierde

Jugador

Reiniciar nivel

Volver al Menú de Niveles

Dentro de un nivel

Sistema

Cargar nivel

Inicializar todo

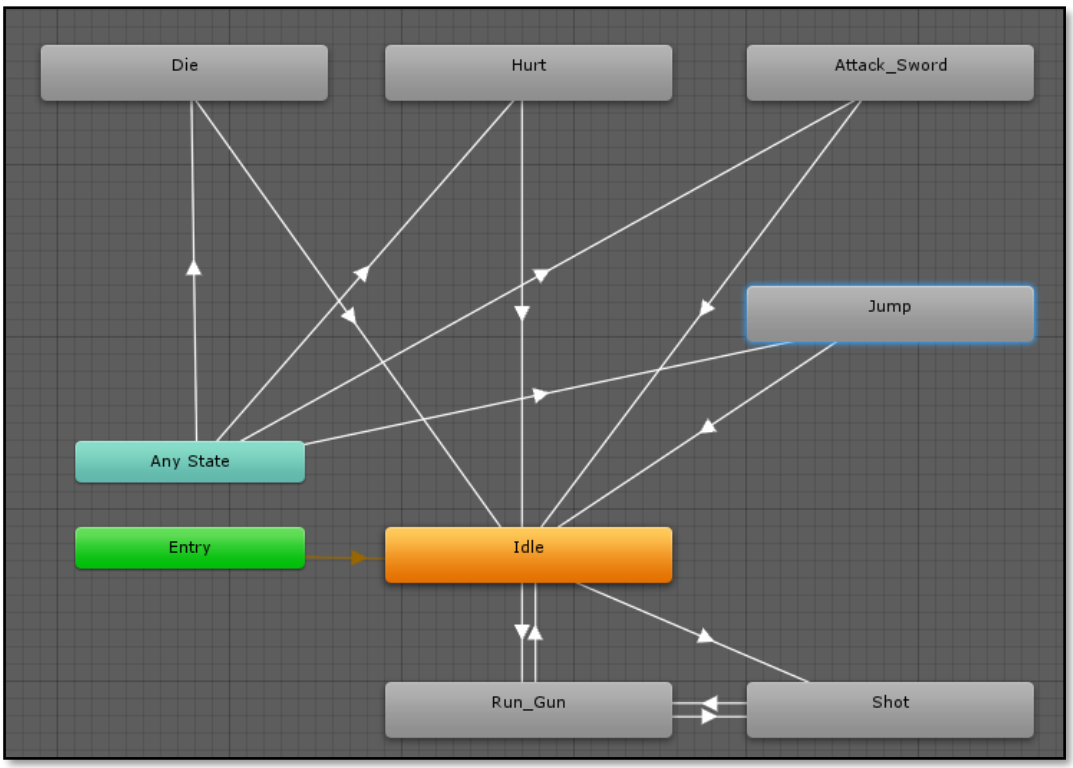
- **Elementos de mayor complejidad:**

Alguno de los elementos que me he encontrado por el camino, han sido más dificultosos que otros, como, por ejemplo, el comportamiento del personaje que maneja el usuario final.

Una de las razones es porque es lo que tiene más código de programación, por tanto, solo por eso ya tiene que ser más complejo, pero otra razón por la que lo hace más complejo es la máquina de estados que tiene el jugador que he tenido que aprender por mi cuenta.

Una máquina de estados en Unity tiene la idea básica de que el personaje que la lleva está vinculado en alguna acción en particular en cualquier momento, es decir, son los diferentes estados, las diferentes animaciones que ese personaje puede tener. En el GDD (Game Design Document) que aparece más adelante, explico las animaciones que tiene el personaje Pink. Esas animaciones es el estado o comportamiento que puede tener en un determinado momento, y ese cambio de estados es lo que tienes que programar para cualquier situación que requiera cierto estado.

El personaje tiene que tener un componente llamado **Animator Controller** que es el que le dice que ese archivo que tú le des, es su máquina de estados, es decir, en ese archivo tenemos que decirle todos los estados/animaciones que puede hacer. En la siguiente imagen representa la máquina de estados de Pink en un nivel horizontal.



Todas las máquinas de estado empiezan por defecto con:

- Entry.
- Any State.

El nodo Entry es para cuando nada más ejecutemos nuestro juego, pasará al estado que tenga asociado, es el primero que se ejecuta. En mi caso, se ejecutará la animación/estado ***Idle*** que es la animación de cuando el personaje está sin realizar ninguna acción para que al usuario final le transmita una sensación de que está en marcha el juego y no está funcionando mal.

Any State es cuando, por ejemplo, la animación de que el usuario esté fallecido por alguna causa, se ejecutará este estado después de cualquier estado que estaba, es decir, si se reproduce la animación de muerte da igual si estará saltando, disparando, etc. que va a ejecutarse este estado.

- **Retos tecnológicos:**

Un reto tecnológico que se me ha presentado es el hecho de que, al tener un Smartphone con el sistema operativo de iOS, necesitamos tener una cuenta de desarrollador de Apple antes de pagar para poder subir nuestro juego a la AppStore.

Después de tener la cuenta, necesariamente necesitamos un ordenador de la marca Apple para poder instalar el juego en el dispositivo móvil a través del programa xCode.

Esto ha hecho un poco más complicado el poder probar el juego, pero al intentarlo en varias ocasiones al final sí que se ha podido realizar. El otro problema que podemos detectar es que cuando te instalas el juego en tu dispositivo, este solo dura unos días activo, llega un momento en que esa versión que instalaste del juego no funciona, por tanto, debemos de volver a instalar el juego como hemos dicho antes, pero al menos podemos probar como funciona. Este problema no ocurriría si se pagara a la marca Apple para poder publicar tus aplicaciones en la AppStore.

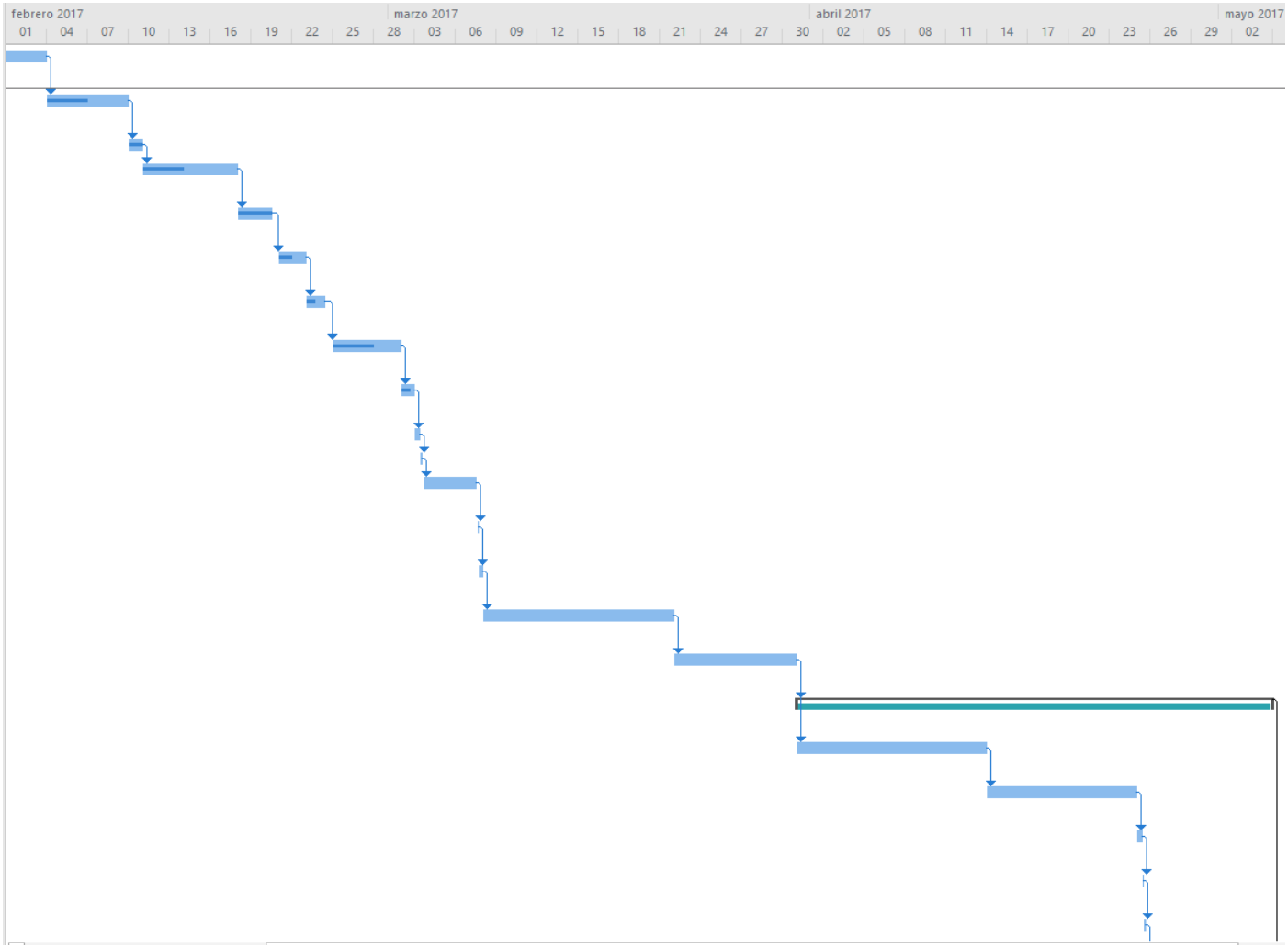
- **Workflow:**

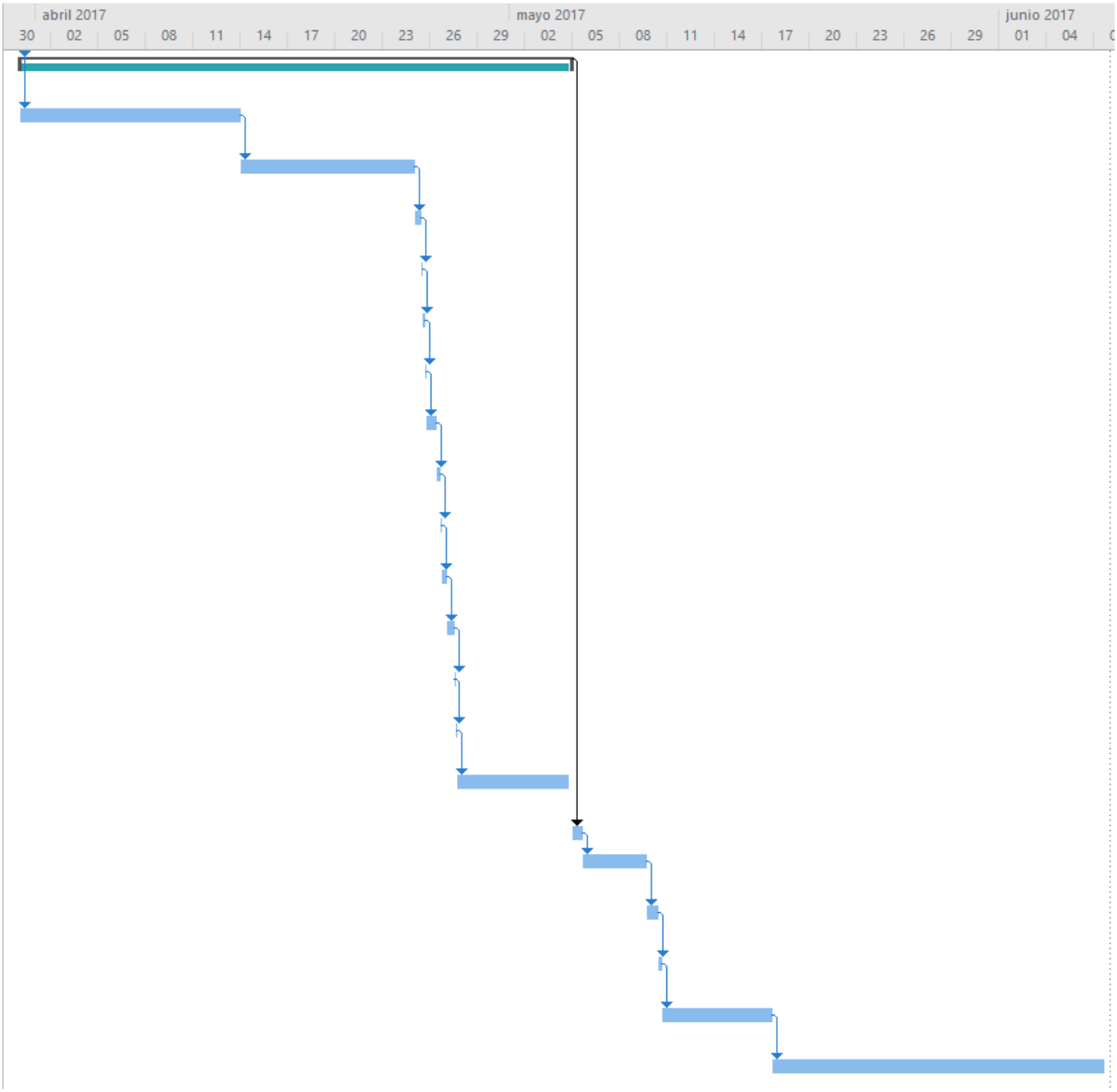
A continuación, muestro todas las tareas que he llevado a cabo para elaborar el proyecto:

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		Diseñar mecánicas del jugador	3 días	mié 01/02/17	vie 03/02/17	
2		Diseñar la estructura del nivel	4 días	sáb 04/02/17	jue 09/02/17	1
3		Diseñar enemigos	1 día	vie 10/02/17	vie 10/02/17	2
4		Diseñar nivel concreto	5 días	sáb 11/02/17	vie 17/02/17	3
5		Diseñar sistema de vida del jugador	3 horas	sáb 18/02/17	lun 20/02/17	4
6		Diseñar menús que tendrá el juego	2 días	mar 21/02/17	mié 22/02/17	5
7		Diseñar items que contendrá el juego	10 horas	jue 23/02/17	vie 24/02/17	6
8		Diseñar los tipos de plataformas que tendrá el nivel	3 días	sáb 25/02/17	mié 01/03/17	7
9		Diseñar la interfaz del jugador en el nivel	8 horas	jue 02/03/17	jue 02/03/17	8
10		Diseñar objetivo del juego	1 hora	vie 03/03/17	vie 03/03/17	9
11		Diseñar ayudas al usuario (Valentina)	3 horas	vie 03/03/17	vie 03/03/17	10
12		Escoger diseños para el entorno	2 días	vie 03/03/17	mar 07/03/17	11
13		Diseñar el incentivo para que el jugador haga el 100% del juego	1 hora	mar 07/03/17	mar 07/03/17	12
14		Diseñar sonidos/música del juego	3 horas	mar 07/03/17	mar 07/03/17	13

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
15		Documentar el Game Design Document (GDD)	10 días	mié 08/03/17	mar 21/03/17	14
16		Desarrollar pixel arts para el juego	7 días	mié 22/03/17	jue 30/03/17	15
17		**Desarrollar un nivel**	**25 días**	**vie 31/03/17**	**jue 04/05/17**	**16**
18		Programar mecánicas jugador	10 días	vie 31/03/17	jue 13/04/17	16
19		Programar plataformas del nivel	7 días	vie 14/04/17	lun 24/04/17	18
20		Programar pick ups	1 hora	mar 25/04/17	mar 25/04/17	19
21		Programar cofres	2 horas	mar 25/04/17	mar 25/04/17	20
22		Programar cajas	2 horas	mar 25/04/17	mar 25/04/17	21
23		Programar sistema de vida del jugador	1 hora	mar 25/04/17	mar 25/04/17	22
24		Desarrollar menú de pausa	3 horas	mar 25/04/17	mié 26/04/17	23
25		Desarrollar menú de niveles	4 horas	mié 26/04/17	mié 26/04/17	24
26		Programar monedas	1 hora	mié 26/04/17	mié 26/04/17	25

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
27		Programar checkpoint	2 horas	mié 26/04/17	mié 26/04/17	26
28		Desarrollar sonidos/música	3 horas	jue 27/04/17	jue 27/04/17	27
29		Programar animales	1 hora	jue 27/04/17	jue 27/04/17	28
30		Programar a Valentina	2 horas	jue 27/04/17	jue 27/04/17	29
31		Decorar el nivel	5 días	jue 27/04/17	jue 04/05/17	30
32		Testear juego	5 horas	vie 05/05/17	vie 05/05/17	17
33		Reparar errores encontrados	2 días	vie 05/05/17	mar 09/05/17	32
34		Testear de nuevo	5 horas	mar 09/05/17	mié 10/05/17	33
35		Preparar encuesta	3 horas	mié 10/05/17	mié 10/05/17	34
36		Repartir la encuesta y obtener resultados	5 días	mié 10/05/17	mié 17/05/17	35
37		Redactar memoria	15 días	mié 17/05/17	mié 07/06/17	36





Sobre el Diagrama de Gantt podemos observar que lo más complejo y duradero ha sido el realizar por primera vez un nivel completo, ya que, se debe hacer las mecánicas del jugador, menús, diseño del nivel etc… Pero tampoco quiere decir que sea fácil crear niveles a partir de uno hecho, porque para hacer que cada nivel no tenga errores se debe de probar muchas veces y eso, ocupa tiempo, además de decorar/diseñar el nivel. Esto también es un trabajo largo y entretenido.

- **Arquitectura de la aplicación / diagramas:**

Todas las escenas que aparecen en el juego están explicadas en el anexo del GDD (Game Design Document), pero podemos detallar un poco más que cada escena tiene su canvas correspondiente, porque por ejemplo en el menú principal habrá un canvas con sus respectivos botones e imágenes para que el usuario pueda interactuar con ellos.

En la siguiente imagen se muestra la escena que es el Menú principal del juego y como hemos dicho tendrá sus respectivos botones para poder interactuar, que como se puede observar son: Start, Options, About y Exit.

Además, tiene asociado tres sistemas de partículas que están en constante movimiento al igual que el personaje de Pink para dar sensación de que el juego está en marcha y no dar la sensación de que ha dejado de funcionar.

Al pulsar el botón Start nos dirigiremos al menú de niveles para poder escoger el nivel que queramos jugar, por tanto, tendremos otro canvas especializado para esta escena y cada nivel tendrá sus respectivas estrellas que serán las que indicará al usuario se ha obtenido el 100% del nivel o no.

Para saber calcular el 100% de un nivel, es decir, si ha destruido todos los enemigos que haya en el nivel, si ha recogido todas las monedas que haya y si ha recogido mínimo 3 diamantes, he hecho el uso de un script denominado Game Manager.

Es una clase pública estática, cuyo objetivo es crear diferentes variables que se puedan acceder a ellas desde cualquier otro script y así poder modificarlas cuando y donde quiera que esté.

4. CONCLUSIONES

- **Capturas de la aplicación:**

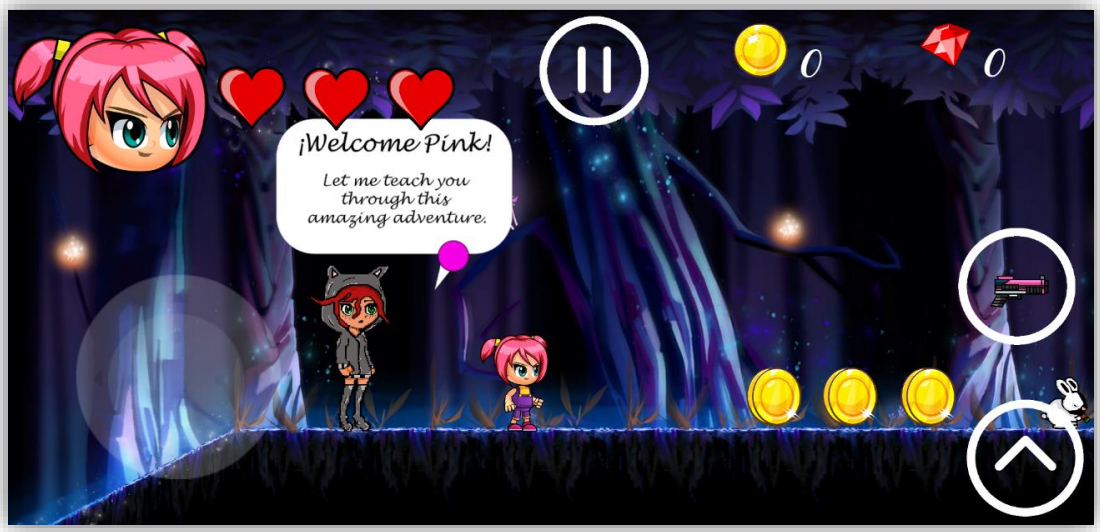
A continuación, se muestran imágenes de la aplicación menos la del menú principal, ya que se muestra anteriormente.

- Opciones:

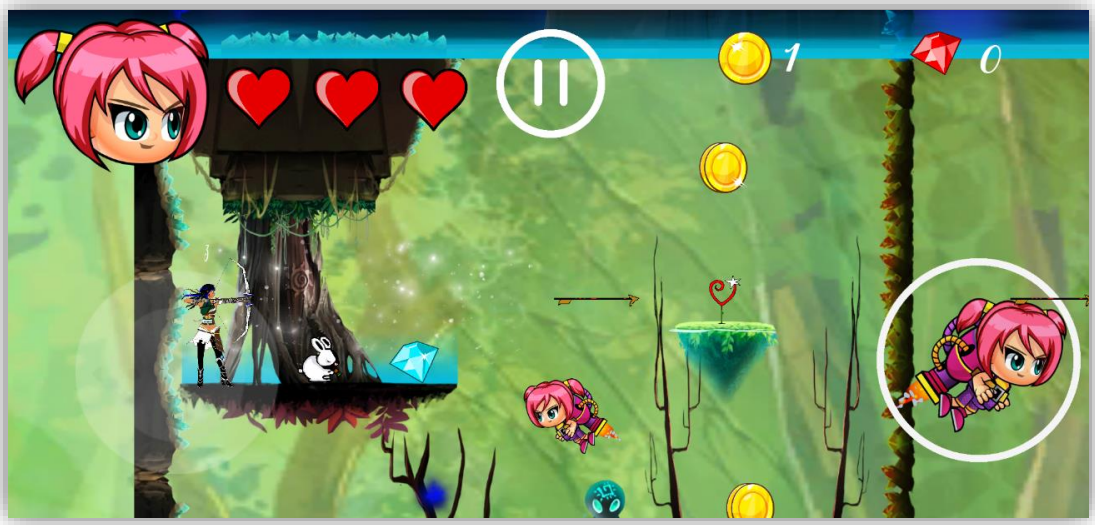
- Acerca de:

- Menú de niveles:

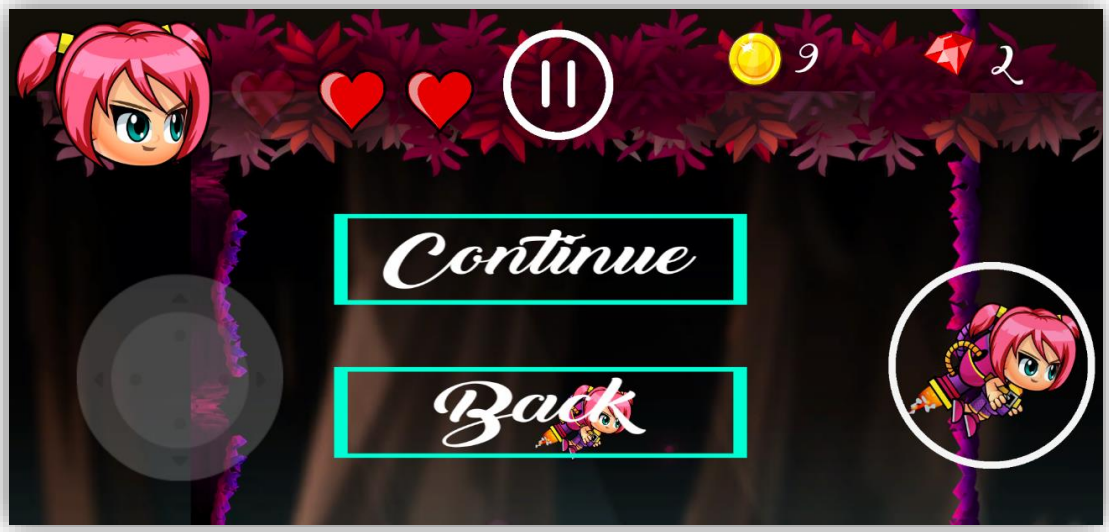
- Interfaz dentro de un nivel horizontal:



- Interfaz dentro de un nivel vertical:



- Menú de pausa/ finalización de nivel/ perder la partida:



- **Publicar en stores:**

He publicado mi aplicación en la tienda de Google que se denomina Google Play, ya que, lo bueno de publicarlo en este lugar es que hay muchísimos dispositivos móviles con sistema operativo Android en el mercado y el precio es más asequible.

A la hora de realizar esta publicación he podido observar todo el trabajo que hay detrás de todo esto, ya que para que los desarrolladores puedan ver todo el seguimiento de su aplicación y para poder configurar o publicar, tienen una plataforma web llamada Google Play Console y aquí es donde te das cuenta de toda la de información que pueden llegar a transmitirte, a esto me refiero a que te dicen cuántas descargas ha tenido tu aplicación, e incluso en qué versión de Android, todas las valoraciones, si todavía las personas que se han instalado tu aplicación siguen teniendo tu juego, además de poder promocionar tu aplicación a través de anuncios o ver desde que país han instalado tu aplicación…

En esta misma plataforma, se pueden publicar tus aplicaciones siguiendo una serie de pasos que hay que configurar, como, por ejemplo, la edad que te califican para tu juego.
Al realizar una serie de preguntas formuladas por Google, él te determina la edad mínima recomendada que tiene que tener el jugador, pero para cada país es un número diferente, ya que cada país tiene sus leyes de censura.

Para mi aplicación el resultado final ha sido el siguiente:

- **Encuestas:**

  He realizado una encuesta para poder tener más información acerca de la opinión de los jugadores que han podido probar mi juego antes de publicarlo y así poder tener una idea de las mejoras que se podrían realizar en un futuro.

  Como la encuesta es anónima, he preguntado algunas cosas personales para poder ver si por ejemplo tiene estudios relacionados con los videojuegos y, por tanto, sabe más del tema del desarrollo, ya que no es lo mismo una opinión subjetiva a una opinión que ya ha pasado por aquí y sabe cómo va todo.

Estas han sido las cuestiones que se han planteado:

- *Sobre la interfaz del juego en un nivel, ¿te ha parecido intuitiva?*
- *¿Cambiarías alguna cosa de la interfaz?*
- *¿Los controles del juego te parecen cómodos y correctos?*
- *Sobre la decoración del juego, ¿qué te ha parecido y que sensaciones te ha transmitido?*
- *Sobre el nivel de dificultad, ¿te ha resultado demasiado fácil/difícil poder completar el nivel al 100%? ¿Por qué?*
- *¿Eres jugador habitual?*
- *Define en tres palabras el juego Pink justificando tu respuesta.*
- *Del 1 al 10, ¿qué nota le pondrías?*
- *¿Por qué crees que se merece esa nota? (respuesta a la anterior pregunta).*
- *¿Qué edad tienes?*
- *¿Sexo?*
- *¿Tienes algún conocimiento sobre la creación de videojuegos?*
- *¿Las animaciones de todos los personajes te han parecido nítidas/correctas? ¿Por qué?*
- *Sobre el objetivo del juego, ¿qué te ha parecido?*
- *¿Crees que es conveniente poner a un personaje que te explique cómo va el juego? (Valentina).*
- *¿Qué cosas propondrías para mejorar el juego?*
- *¿Crees que este juego en otras plataformas como PC o consolas estaría bien?*
- *¿Qué género de videojuegos te suele gustar más?*
- *¿Has desarrollado algún videojuego o piensas hacerlo en un futuro?*
- *Sobre la música y sonidos, ¿qué te han parecido? Justifica tu respuesta.*
- *¿En qué te sueles fijar en el momento que ves un videojuego por primera vez?*

Los resultados obtenidos los voy a dividir en tres bloques, que son los siguientes:

- ***Resultados*** de personas ***con algún conocimiento*** sobre videojuegos.
- ***Resultados*** de personas que ***no tienen ningún conocimiento*** sobre videojuegos.
- ***Resultados generales*** sin tener en cuenta ninguna condición anterior.

1. Resultados con conocimientos previos:

¿Interfaz intuitiva?	¿Controles correctos?	¿Estéticamente agradable?	¿Dificultad?	¿Animaciones correctas?	Descripción objetivo
100% Si	60% Sí 40 % No	100% Si	20% Difícil 20% Normal 60% Fácil	100% Si	60% Entretenido 40% Sencillo

¿Necesidad de ayuda (Valentina)?	¿PC o Consola?	¿Música/Sonidos adecuados?	Nota individual sobre 10	Nota final sobre 10
80% Si 20% No	100% Si	100% Si	9 7 6 10 9	8,2

Resultados obtenidos al haber realizado la encuesta a 5 personas, de las cuales 3 son hombres y 2 mujeres.

2. Resultados sin conocimientos previos:

¿Interfaz intuitiva?	¿Controles correctos?	¿Estéticamente agradable?	¿Dificultad?	¿Animaciones correctas?	Descripción objetivo
100% Si	100% Si	80% Si 20% No	60% Difícil 40% Normal	100% Si	60% Práctico 40% Sencillo

¿Necesidad de ayuda (Valentina)?	¿PC o Consola?	¿Música/Sonidos adecuados?	Nota individual sobre 10	Nota final sobre 10
100% Si	60% No 40% Sí	100% Si	8 7 10 8 9	8,4

Resultados obtenidos al haber realizado la encuesta a 5 personas, de las cuales 2 son hombres y 3 mujeres.

Las diferencias entre el primer resultado y este, no es que sean muy significativas, pero la que más resalta es que quien ha estudiado y tiene idea de estos conocimientos sí que piensan que podría ser un buen mercado el publicarlo para PC o Consola, esto puede deberse a que los que suelen ser jugadores habituales también les gusta jugar en todas las plataformas y por eso conlleva a querer tener conocimientos sobre este tema.

Además de que al haber jugado más, el porcentaje de dificultad varía significativamente, ya que quien no suele jugar, no tiene experiencia de ningún tipo y, por tanto, le resultará más complejo.

3. Resultado general:

¿Interfaz intuitiva?	**¿Controles correctos?**	**¿Estéticamente agradable?**	**¿Dificultad?**	**¿Animaciones correctas?**	**Descripción objetivo**
100% Si	80% Si 20% No	90% Si 10% No	40% Difícil 30% Normal 30% Fácil	100% Si	40% Entretenido 30% Práctico 30% Sencillo

¿Necesidad de ayuda (Valentina)?	**¿PC o Consola?**	**¿Música/Sonidos adecuados?**	**Nota final sobre 10**
90% Si 10% No	70% No 30% Sí	100% Si	8,3

Estos resultados se han obtenido al sacar las conclusiones de las 10 personas juntas.

Cabe añadir que, para describir el juego en tres palabras, las más usadas han sido:

- ***Divertido.***
- ***Entretenido.***
- ***Amigable.***

Pienso que son tres palabras que me producen mucha satisfacción, ya que parece que he conseguido que a mi juego les guste por lo menos a 10 personas que he cuestionado.

Las edades a las que se ha hecho dicha encuesta han sido desde 19 años hasta 36 años, donde han participado 5 hombres y 5 mujeres.
Por tanto, he de aclarar que me ha sido útil el hecho de hacer estas valoraciones, porque ha servido para aprender los pequeños fallos y poder tener más visión de futuro para próximas actualizaciones que hablaré en el siguiente punto de la memoria.

- **Trabajo futuro:**

Después de haber preguntado con las encuestas, he de decir que había conceptos que no había pensado antes, pero que ahora me han ayudado para próximos proyectos futuros, como, por ejemplo, el hecho de que, en el primer nivel, podría haber explicado mejor con algún tutorial que hiciera que el jugador se enseñara los controles paso a paso, además de tener al personaje de Valentina para facilitar el aprendizaje del juego al jugador.

En futuras actualizaciones me gustaría publicar el juego para las plataformas de PC o Consola, ya que pienso que podría ser un gran mercado para darte a conocer, pero antes, me gustaría el poder añadir más idiomas para la aplicación, porque ahora mismo solo está disponible en inglés y al menos, me gustaría añadir el español.

Por último, añadiría algún objeto que el jugador pudiera coger en niveles y le transformara para poder tener poderes especiales, por ejemplo, siendo inmune a los ataques de los enemigos durante un tiempo, o poder tener diferentes balas cuando dispara con la pistola para poder hacer más o menos daño a los enemigos, etc.

Se pueden hacer muchas cosas todavía, pero creo que con el tiempo dedicado y esfuerzo el resultado ha sido muy satisfactorio, espero poder añadir más niveles al juego con diferentes mecánicas de ahora en adelante para poder desarrollar aún más el potencial que puede llegar a tener este juego.

5. ANEXOS

- **GDD:**

A continuación, muestro el documento que todo juego tiene que hacer antes de empezar su fase de desarrollo que se denomina **Game Design Document**, es decir, documento de diseño del juego.

Es un documento donde se explica todo lo que tiene que tener el juego al finalizar su desarrollo, por tanto, se tienen que pensar todas las cosas que queremos de él y plasmarlas en este documento.

Por decirlo de otra manera, cualquier persona que lea este documento y no tenga nada que ver con dicho juego, tiene que ser capaz de describir todo lo que tiene.

GAME DESIGN DOCUMENT

PINK

Mariola Trenzano Álvarez

ÍNDICE

INTRODUCCIÓN

Descripción del Proyecto

Pink es un juego plataformas donde el objetivo del jugador es encontrar a través de la exploración del nivel, 3 gemas para poder completarlo y seguir jugando.

Nuestra protagonista llamada Pink, se encuentra en un lugar que desconoce denominado bosque encantado. El objetivo de este juego es conseguir todo lo que se pueda para salir de allí con vida.

Descripción Técnica

El juego se está desarrollando íntegramente en el motor Unity y la plataforma a la que va a ir diseñada es para Smartphone Android.

DISEÑO DEL JUEGO

Gameplay General y Mecánicas Principales

La protagonista avanza por el nivel luchando contra enemigos y el nivel finaliza cuando se cumplen uno de los dos acontecimientos siguientes:

- Cuando llega al final del nivel y tiene 3 gemas recogidas siendo un nivel horizontal. En los niveles verticales solo tiene que llegar al final del nivel.
- Cuando el personaje muere, ya sea cayendo al vacío, cayendo en unos pinchos o sea eliminado por algún enemigo.

Se completa el nivel cuando el personaje llega a un cierto punto del nivel y habiendo cumplido sus objetivos. Al entrar en contacto con ese punto, se volverá al menú donde están todos los niveles para poder pasar al siguiente o si quiere volver a repetir el mismo.

El personaje puede perder por tres razones:

- Pierde cuando su vida tiene 0 corazones.
- Pierde cuando cae al vacío.
- Pierde cuando cae en unos pinchos.

El nivel tendrá uno/ dos o ningún checkpoints para que el jugador pueda hacer respawn en ese mismo sitio donde esté en el caso de que pierda en algún momento del nivel. El sitio donde aparecerá el jugador será en el último checkpoint que habrá recogido simplemente con tocarlo.

Objetivos

El objetivo del jugador para finalizar el nivel debe ser encontrar las 3 gemas que estarán esparcidas por dicho nivel en el que se encuentre y con ellas llegar al punto donde finalice éste.

Si el objetivo del jugador es completar el nivel al 100% tendrá que recoger, aparte de las 3 gemas, todas las monedas y derrotar a los enemigos que haya por el nivel.

JUGADOR

La vida del jugador cuenta con 3 corazones, por tanto, dependiendo del enemigo le quitará uno o dos corazones.

Habrá pick ups repartidos por el nivel que sirven para recuperar vida y, por tanto, cada uno de ellos aumentará un corazón de la vida del personaje. Una vez se quede con 0 corazones, el personaje morirá y se hará respawn a algún checkpoint que habrá cogido anteriormente o si no ha cogido ninguno, se empezará de nuevo el nivel.

El jugador cuenta siempre con una pistola en un nivel horizontal, de la cual cuando pulse un botón, podrá disparar una serie de balas durante un tiempo limitado, cuando haya descargado 10 balas, tendrá que esperar unos segundos para que se vuelva a recargar las balas para poder seguir disparando.

Nuestro personaje podrá disparar balas como hemos dicho anteriormente y afectarán a los enemigos que les toque con ellas.

Cuando dispare, las balas se desplazarán en la dirección donde el personaje mire cuando se haya pulsado el botón de disparar, es decir, si el jugador pulsa el botón cuando el personaje mira hacia a la derecha, las balas irán para la derecha.

El jugador puede ser atacado por los enemigos ya sea por el arma del enemigo o por entrar en contacto con él.

La información al jugador aparecerá mediante un personaje que estará en el nivel en forma de mensajes explicando los controles del juego.

El jugador interactuará con las cajas del nivel mediante el mismo botón que dispara para que le resulte más fácil y cómodo los controles, pero en vez de disparar balas, le cambiará el arma por un sable para poder abrir la caja. En ese momento el botón de la interfaz cambiará.

Para poder moverse por el escenario horizontalmente usará un joystick que aparece en pantalla y si pulsa un botón que estará en la interfaz podrá saltar y así podrá moverse verticalmente también.

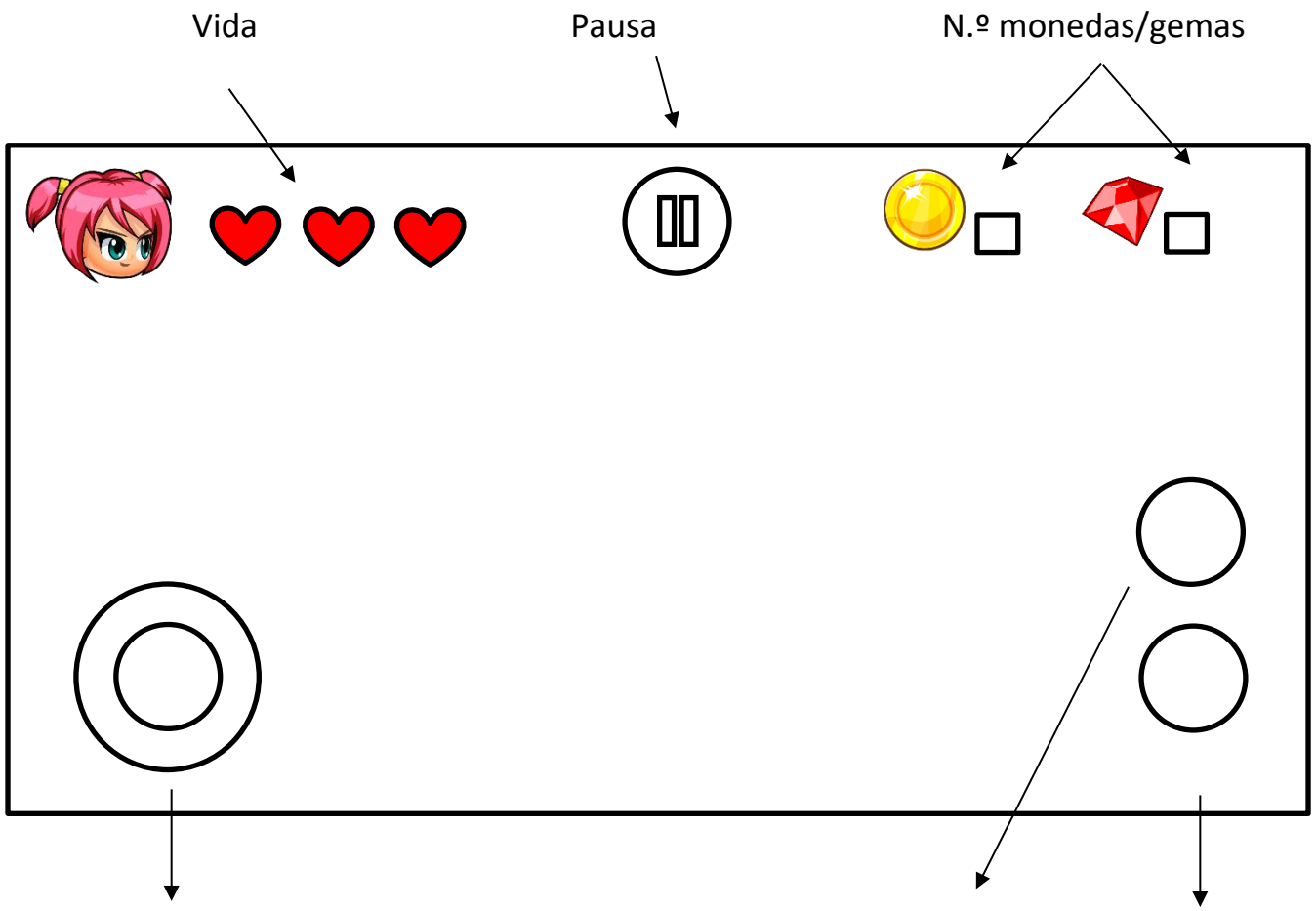
Mientras nuestro jugador esté en el nivel puede poner pausa apretando un botón que aparecerá en la interfaz del nivel. Al pulsarlo, le aparecerá un menú con la opción de volver a empezar el nivel, continuar por donde va o volver al menú donde están todos los niveles para poder elegir otro nivel, si es ese su deseo.

Esto sería cuando están en niveles horizontales, pero en el momento que estén en niveles verticales el jugador tendrá un jetpack en el que simplemente tendrá que ir recogiendo los ítems sin chocar con los pinchos y esquivando las flechas que les dispararán los enemigos para llegar al final del nivel.

Si el personaje muere en un nivel vertical, mostrará una animación de una explosión y tendrá que empezar de nuevo el nivel, ya que en los niveles verticales no habrán checkpoints para hacer una experiencia más divertida y desafiante.

INTERFAZ DE USUARIO

A continuación, muestro la interfaz que tendrá el usuario siempre que esté en un nivel horizontal:

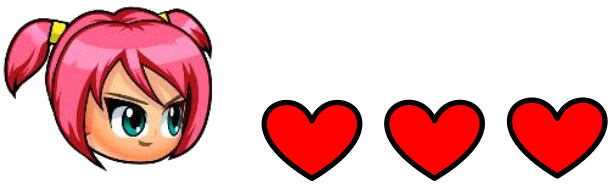


En un nivel vertical, simplemente cambia la parte derecha porque solo tendrá un botón, ya que servirá para darle un impulso hacia arriba a Pink cada vez que el jugador presione el botón.

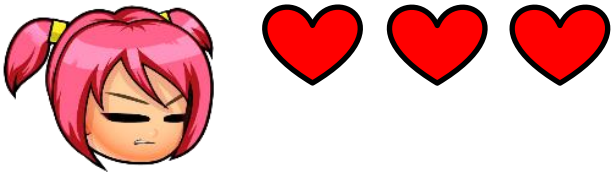
SISTEMAS

Sobre la vida del personaje, habrá una imagen del protagonista justo al lado, esta imagen es la cara de él mismo, que cambiará cuando le hagan daño y volverá a cambiar al estado normal cuando dejen de hacerlo.

Estado normal:

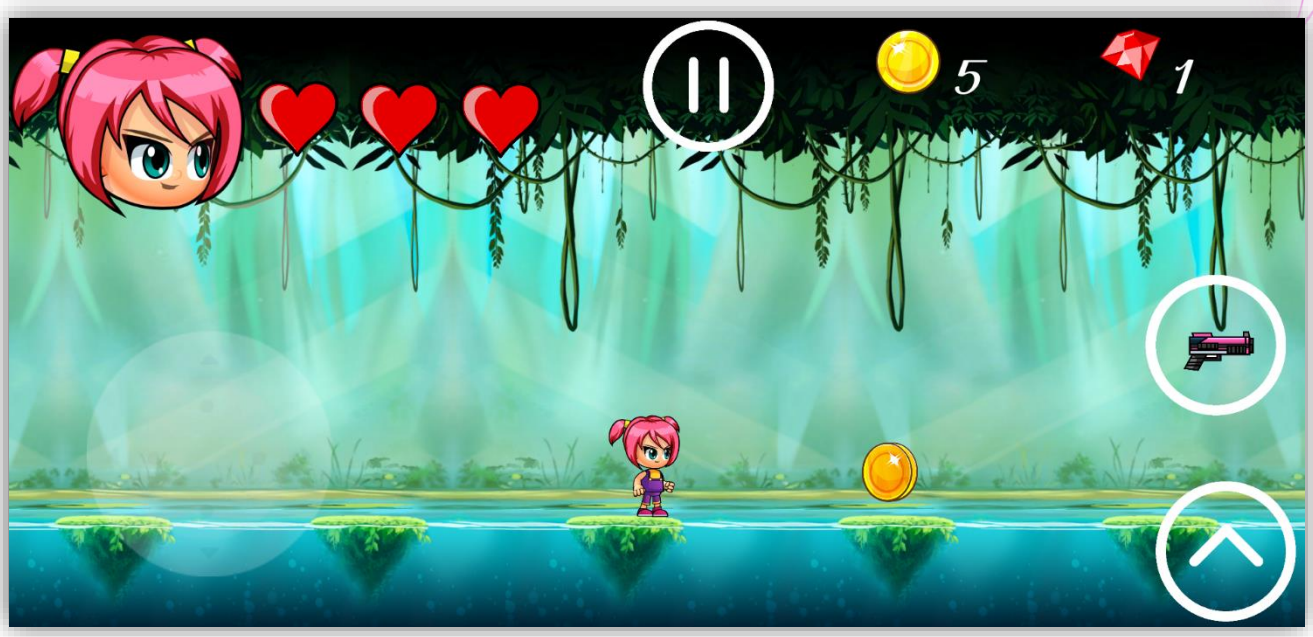


Cuando le hacen daño:

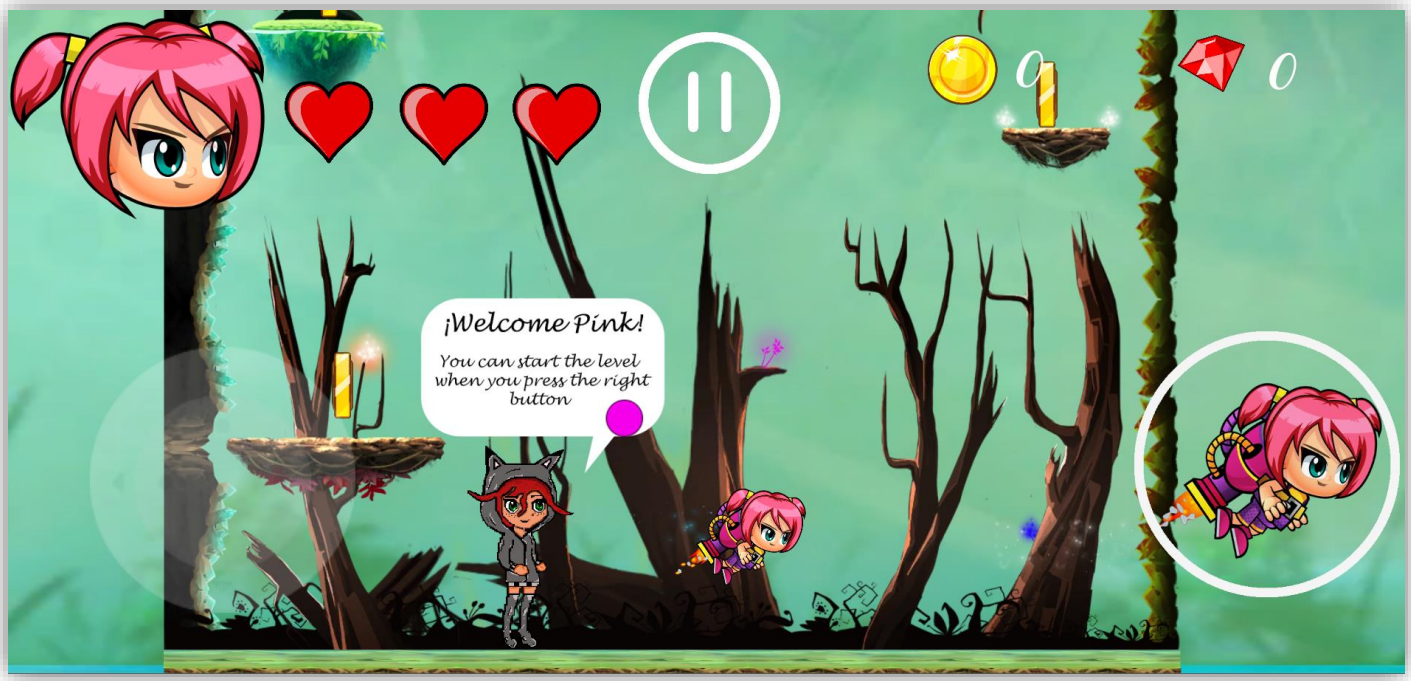


Como hemos visto en la anterior página, el jugador tendrá un contador de monedas y de gemas en la parte superior de la pantalla donde sabrá cuantas monedas y gemas ha recogido. Justo al lado del contador, tenemos el botón de pausa, el cual hemos nombrado anteriormente que es para pausar el nivel en cualquier momento.

En la parte inferior de la pantalla, observamos que a mano izquierda tenemos el joystick donde el jugador puede mover al personaje de izquierda a derecha siempre que quiera. A mano derecha, tenemos dos botones, el de arriba será el botón para poder disparar o para poder atacar con el sable y, por último, tenemos el botón justo abajo que es para poder saltar, que solo se va a poder saltar una vez y será cuando estemos tocando la superficie del suelo.



En el caso de ser un nivel vertical, la interfaz ya no tendrá los botones de disparar y saltar, se sustituirá por un botón que será para iniciar el vuelo con el jetpack y para impulsar a Pink, como se muestra en la siguiente imagen:



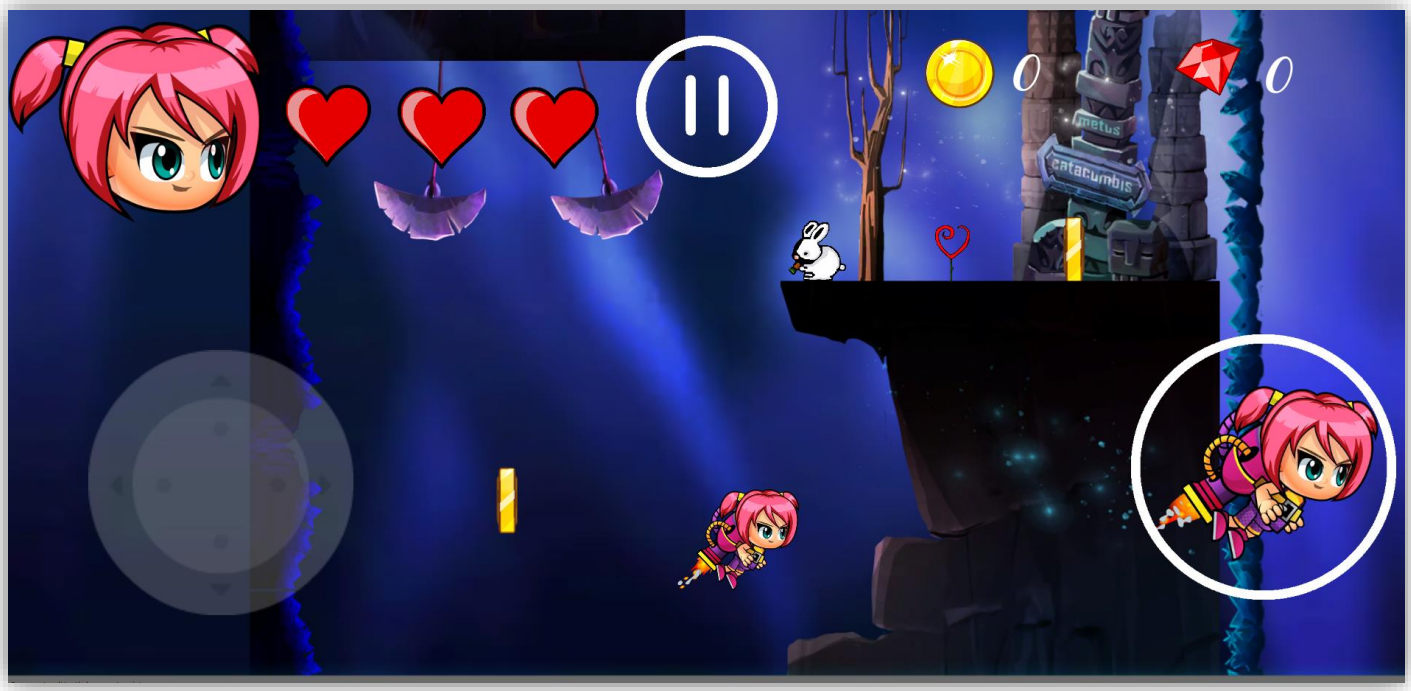
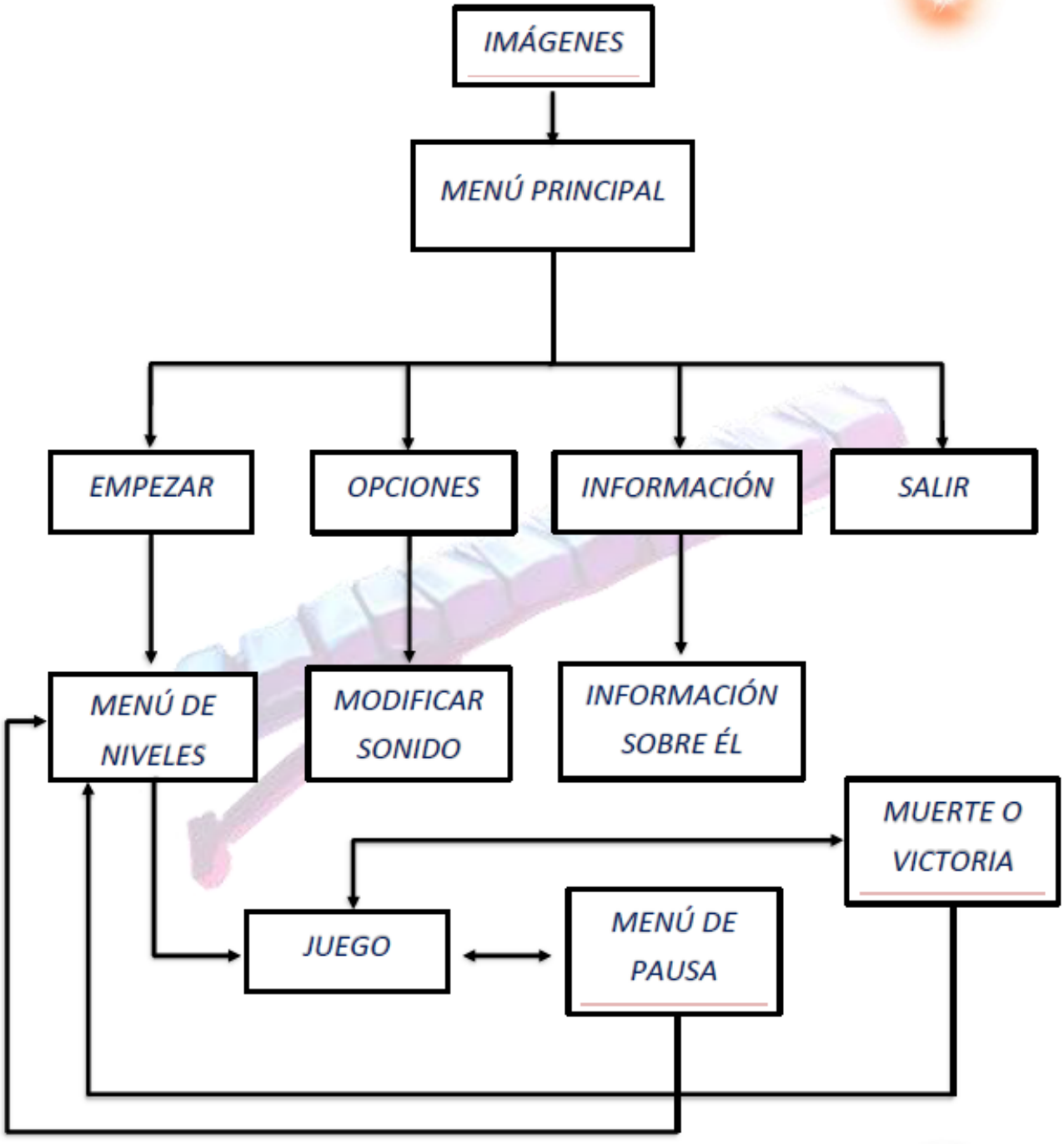


DIAGRAMA DE FLUJO

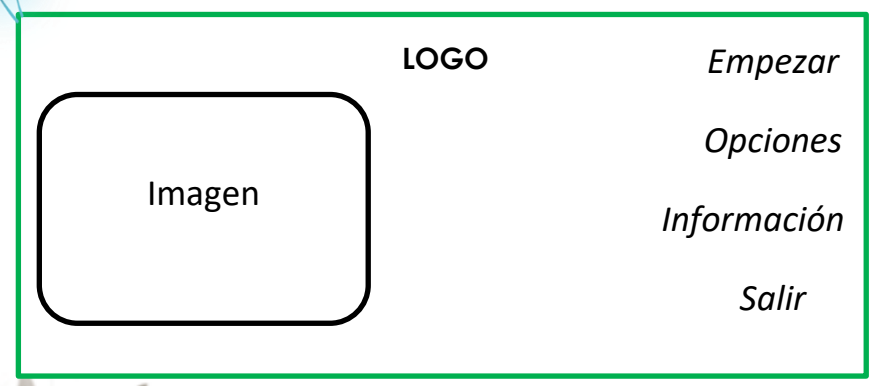


A continuación, explicamos el diagrama de flujo que hemos dado anteriormente.

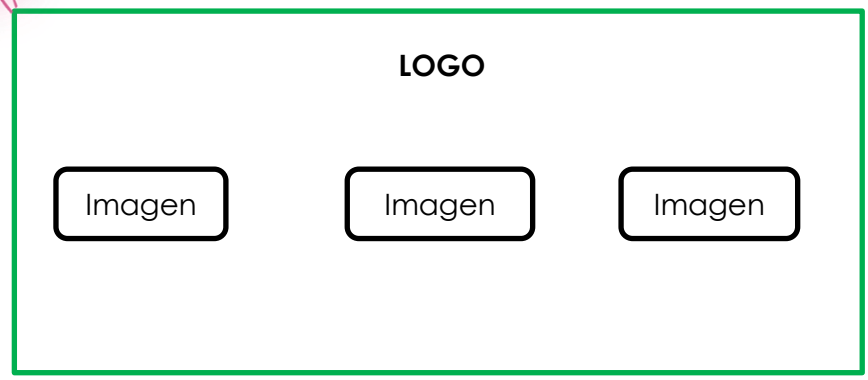
- **Imágenes*:***
  Aparecerá una imagen donde aparecerá mi nombre y que está hecho con el motor Unity.

  Resultado final:

- **Menú principal:**
  El menú principal constará con cuatro botones, los cuales al seleccionarlos nos llevará a un sitio o a otro.
  Los botones son: Empezar, Opciones, Información y Salir.
  Anteriormente ya se ha mostrado la imagen final.



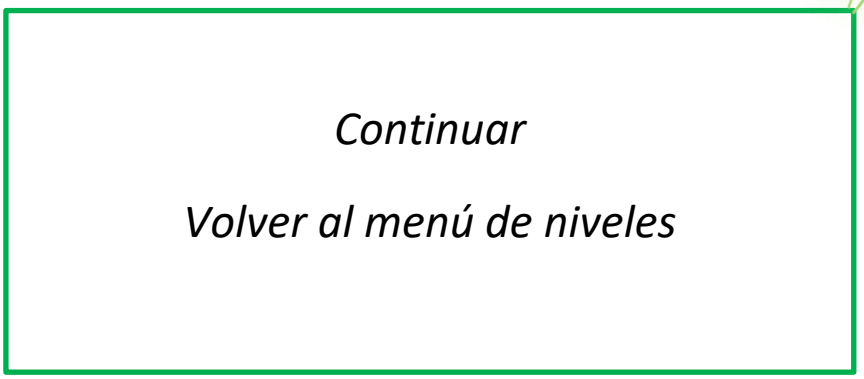
- **Menú de niveles:**
  Aparecerán todos los niveles de nuestro juego para poder elegir el nivel que queramos jugar.
  Anteriormente ya se ha mostrado la imagen final.



- **Empezar:**
  Al pulsar este botón nos dirigirá a la pantalla de Menú de niveles.

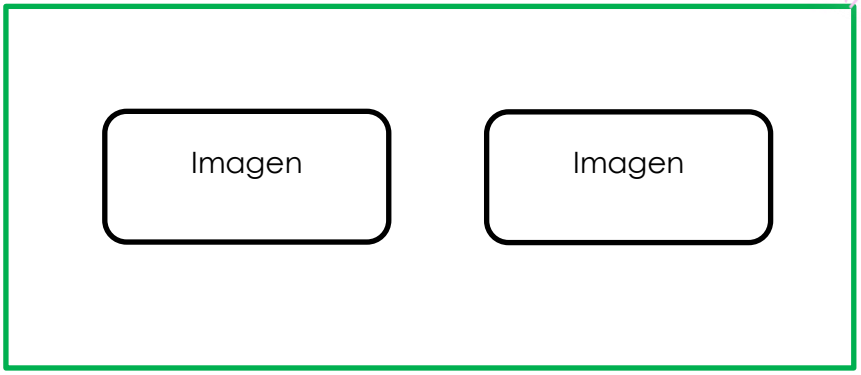
- **Juego:**
  Se refiere al gameplay del juego, es decir, estamos dentro de un nivel del juego, cuya interfaz está descrita anteriormente.

- **Menú de pausa:**
  Durante el juego, podemos poner pausa cuando queramos y tenemos dos opciones, volver al nivel por donde lo habíamos dejado o volver al Menú de niveles.
  Anteriormente ya se ha mostrado la imagen final.



- **Opciones:**
  Al pulsar este botón nos dirigirá a la pantalla de Sonido.

- **Sonido:**
  Aquí podremos modificar el sonido y la música del juego.
  Anteriormente ya se ha mostrado la imagen final.



- **Información:**
  Al pulsar este botón nos dirigirá a la pantalla de Información sobre él.

- **Información sobre él:**
  Aparecerá información acerca del juego.

- **Muerte o Victoria:**
  Aparecerá una pantalla donde las opciones será reiniciar el nivel si ha muerto o si vuelve al menú de niveles si ha ganado.

- **Salir:**
  Al pulsar este botón podremos salir del juego.

ENEMIGOS

TROLL

Habrá enemigos de este tipo esparcidos por ciertos puntos del nivel.

Estos, se moverán de un punto A hacia un punto B mientras no detecte la presencia del jugador.

Para poder detectar al jugador, los enemigos tendrán un rango de distancia, en cuanto detecten que están dentro de ese rango, se pondrá a atacar con un hacha que tiene, si esa hacha o él mismo toca al jugador, le restará un corazón de vida.

La vida de los enemigos serán 30 puntos, de los cuales, se podrán ir restando hasta destruirlo gracias a las balas que le golpeen lanzadas por el protagonista.

ARCHER

Habrá enemigos de este tipo esparcidos por ciertos puntos del nivel.

Estos, permanecerán quietos en ciertos puntos del nivel y cuando detecte la presencia del protagonista le disparará una flecha cada cierto tiempo que le quitará 2 corazones al jugador si lo tocan.

La vida de estos enemigos será de 3 puntos y se podrán eliminar con las balas o con el sable del protagonista.

Este tipo de enemigo representa a las dueñas del bosque en el que se encuentra el protagonista, por eso intentan destruirle y mandan a los Trolls para defender y proteger lo que es suyo.

DISEÑO DE NIVELES

Descripción General de los Niveles

Los niveles van a tener muchos tipos de plataformas que se detallan más adelante, constará de ítems como, por ejemplo, gemas, monedas, vida, cofres y cajas, de los cuales también se describirá todo a continuación.

Cada nivel constará de X número de monedas y 3 gemas como mínimo siempre, los demás ítems, se desarrollará en cada nivel de una manera dependiendo del nivel de dificultad, y de cómo se diseñe dicho nivel.

Los niveles seguirán un patrón en el que el jugador podrá moverse tanto para delante como para atrás del nivel, es decir, que no es un juego que simplemente nos moveremos hacia delante (hacia la derecha) y no podremos volver atrás.

Como se ha dicho anteriormente, para completar el nivel 100% se necesitará recoger todas las monedas, eliminar a los enemigos que haya y recoger las 3 gemas que estarán por el nivel.
En el menú donde aparecen todos los niveles, constará cada uno con 3 estrellas, si se consigue el 100% del nivel, éste se reflejará en esa interfaz con 3 estrellas.

Si solamente se recogen las 3 gemas necesarias para desbloquear el siguiente nivel, el jugador obtendrá solo una estrella.

Si se recoge todas las monedas y las 3 gemas, obtendrá 2 estrellas y, por último, si además de esto elimina a todos los enemigos, como hemos dicho obtendrá las 3 estrellas.

Mecánicas de Juego Asociadas

Las plataformas por las que tendrá que saltar nuestro protagonista en los niveles del juego son las siguientes:

- **Plataformas dañinas:**

Son plataformas que te quitan un corazón cuando el protagonista las toca al subir en ella.

- **Plataformas destructivas:**

Son plataformas que se destruyen a los segundos de que el jugador haya entrado en contacto con ellas.

- **Plataformas movibles:**

Son plataformas que se pueden mover en vertical o horizontalmente.

- **Plataforma 180 grados:**

Son plataformas que tendrán en la parte de debajo de ella pinchos que podrán hacer que el jugador pierda al entrar en contacto con ellos. Cada cierto tiempo la plataforma rotará 180º.

- **Plataforma con rotación:**

Son plataformas que están rotando todo el tiempo y que si el jugador las toca, le quitarán un corazón de vida.

Durante el nivel el jugador podrá encontrarse con los siguientes objetos:

- **Cajas:**

Pueden estar en el nivel y que, al destruirlas con el sable, puede que haya objetos que el personaje pueda recoger, como monedas o gemas.

Tiene una única animación que es la destrucción de la caja.

- **Cofres:**

Son cofres que pueden estar en el nivel y que al abrirlos con solo tocarlos pueda recoger el objeto que hay dentro, que será alguna gema.

Al igual que las cajas, tienen una única animación que será la de abrir el cofre, por tanto, una vez abierto, se quedará abierto durante todo el nivel.

- **Power ups:**

El jugador los recoge tocándolos y pueden estar escondidos en cajas o esparcidos por el nivel. Los power ups pueden ser monedas o gemas.

Las monedas tendrán una animación que será dar vueltas siempre hasta que el jugador la recoja.




Sobre las gemas habrá de tres clases, roja, verde o azul que serán las siguientes:

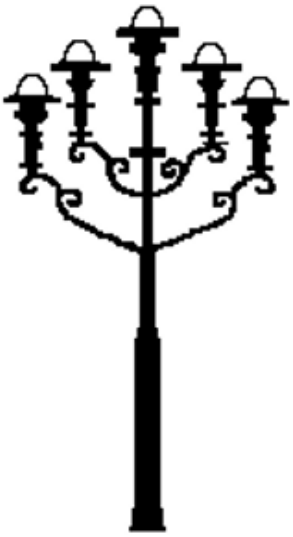
- **Pick ups:**

Lo mismo que con los Power ups, pero éste solo puede ser un corazón de vida. Al igual que las monedas, tendrá una animación que se repetirá en bucle hasta que el jugador lo recoja.

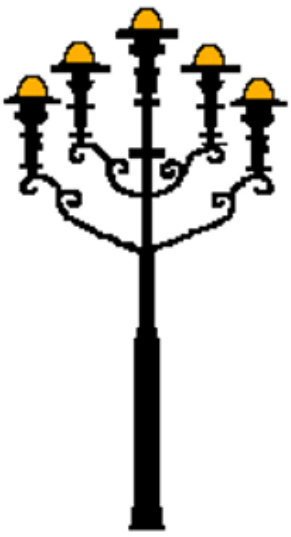
- **Checkpoints:**

Como hemos dicho anteriormente, habrá uno o dos checkpoints para poder hacer respawn si nuestro jugador muere.

Tendrá una animación que se activará cuando el jugador haya tocado/cogido ese checkpoint, para que se dé cuenta que está activo y que, si muere, comenzará desde ese punto de guardado.



Checkpoint desactivado.



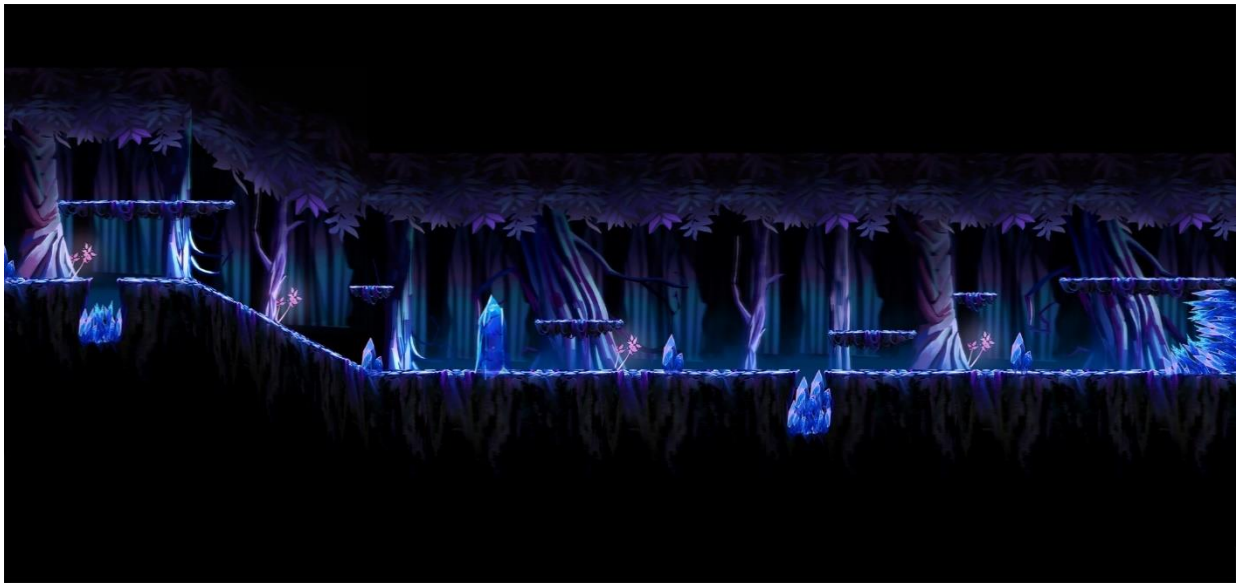
Checkpoint activo.

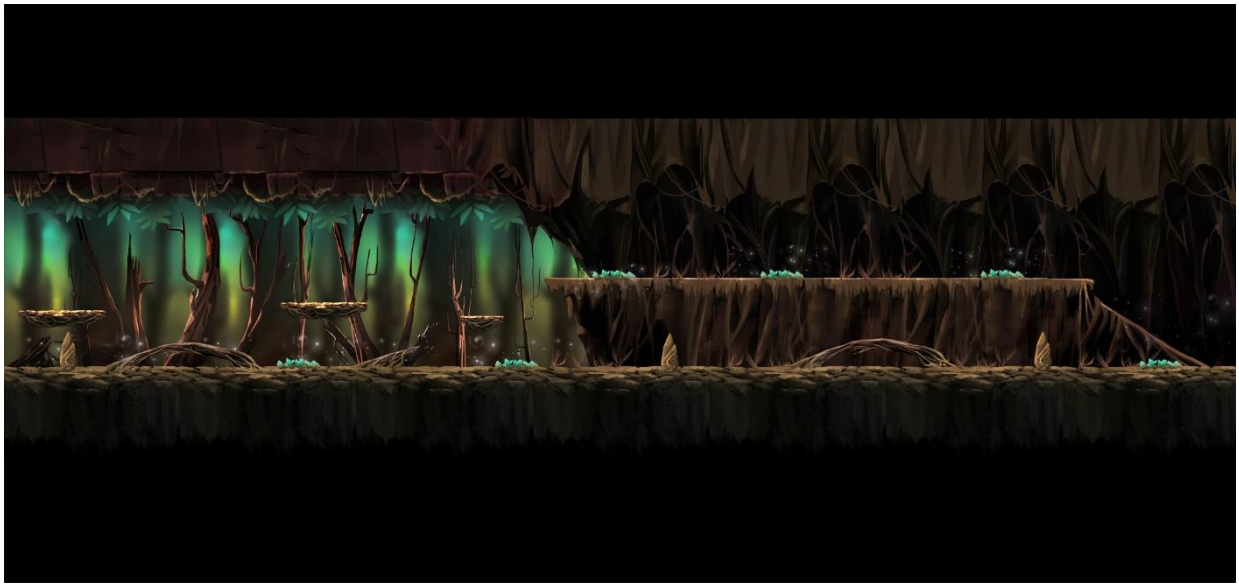
ARTE

Descripción General del Diseño Artístico

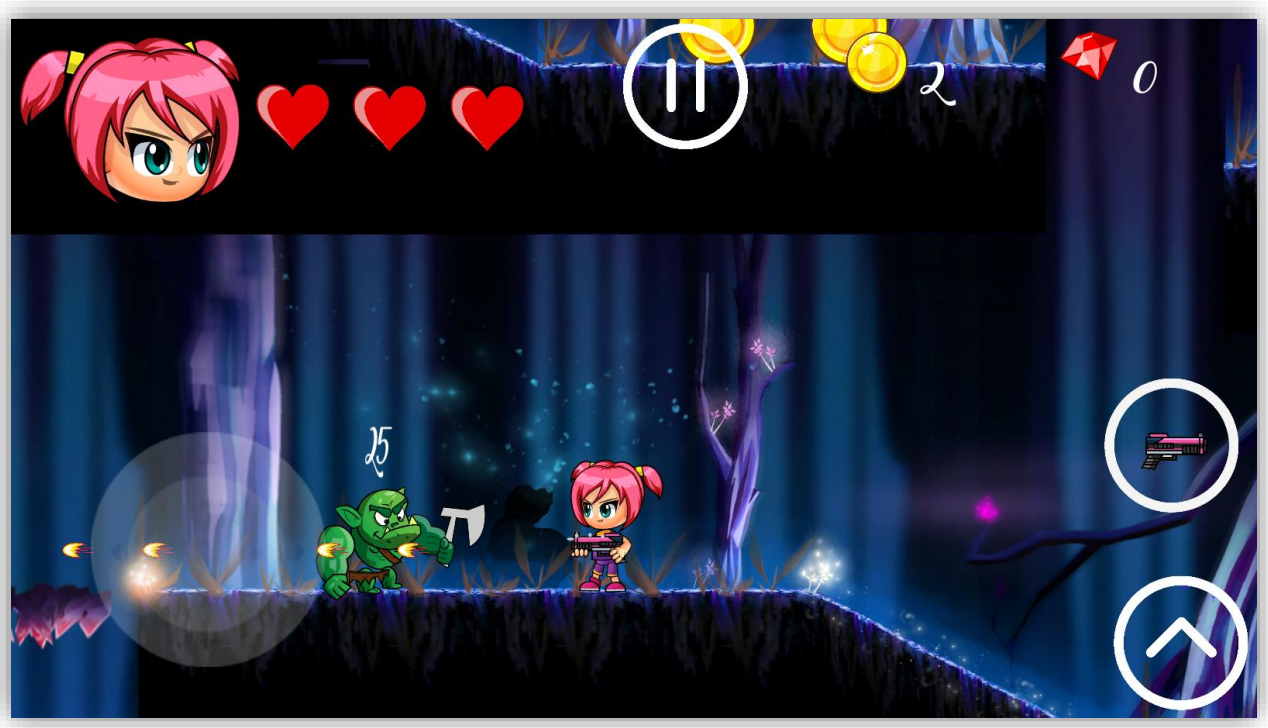
El diseño artístico de este juego tiene un estilo cartoon tanto en el protagonista como en el entorno en el que se encuentra. Tiene colores muy característicos para dar un toque más alegre o para dar un toque más tétrico, dependiendo del nivel, hace que el jugador se pueda sentir más a gusto con el paisaje y querer estar más tiempo, o justo lo contrario y querer pasarse el nivel cuánto antes.

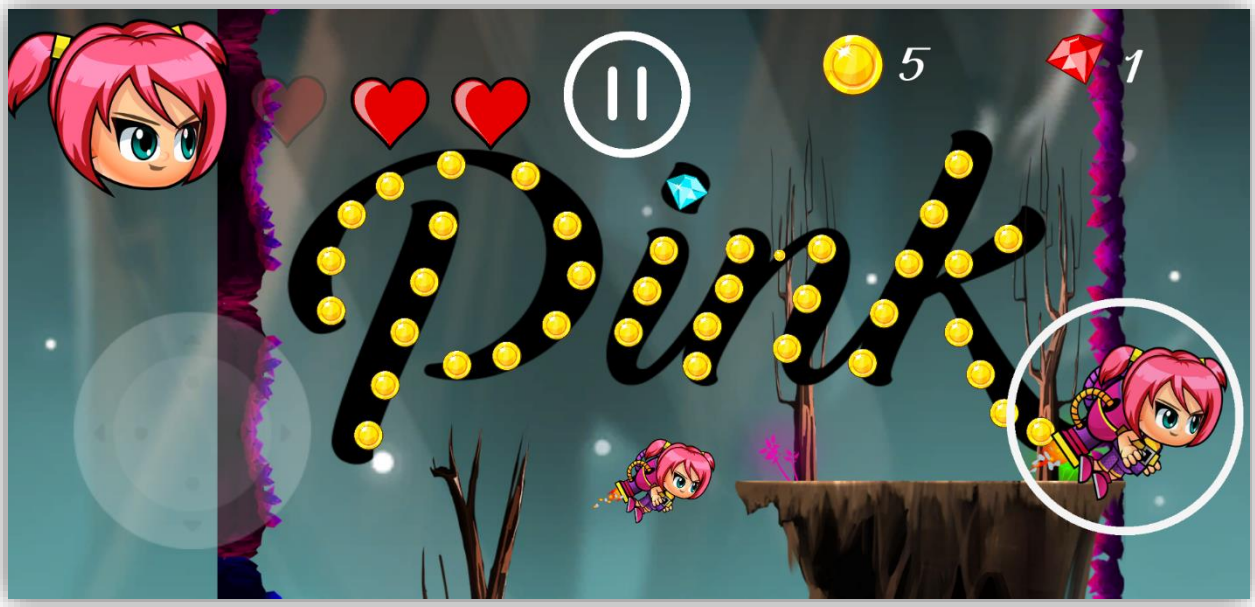
Unos ejemplos del entorno serían:

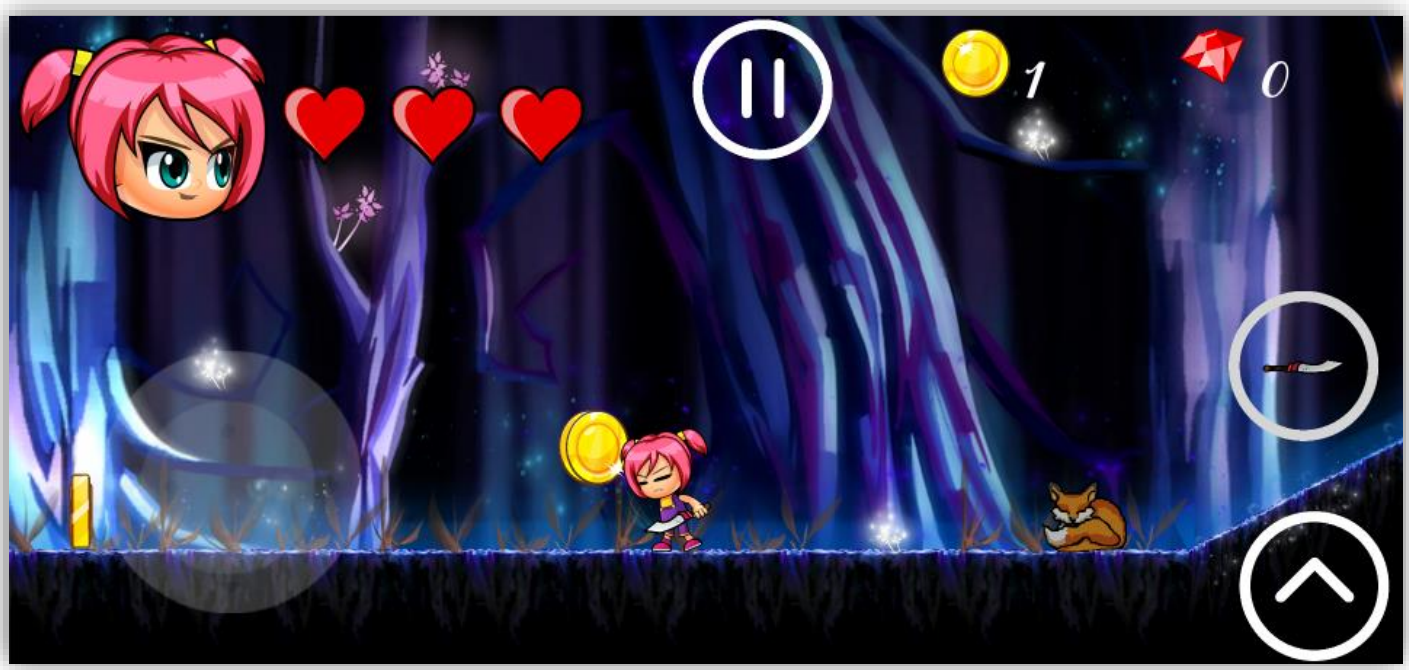




A continuación, se muestran imágenes del juego dentro de un nivel:






1
0


0
0

Al estar dentro de un nivel podemos encontrarnos con estos animales para darle más vida al entorno, además de obtener más satisfacción para la visión de nuestro jugador tendremos la sensación de que el bosque está vivo.

Los animales son conejos y zorros que siempre tendrán una única animación que estará en bucle constante.

- **Conejo:**

- **Zorro:**

PERSONAJES

PINK

Nuestra protagonista llamada Pink, es una niña traviesa que lucha por lo que quiere y para ello, necesita defenderse de los enemigos que se va encontrando con su pistola o con su sable.

Desde el principio de cada nivel, Pink tiene su pistola como arma principal, con la cual, podemos disparar a nuestros enemigos para poder avanzar por el nivel, pero cuando se acerca a una caja, esta arma cambia a ser un sable.

ANIMACIONES

Las animaciones que va a tener este personaje son:

- **Disparar:**

Animación que surge cuando pulsamos el botón siempre y cuando tengamos balas y estamos lejos de las cajas.

- **Cortar:**

Animación que surge cuando estamos cerca de alguna caja.

- **Saltar:**

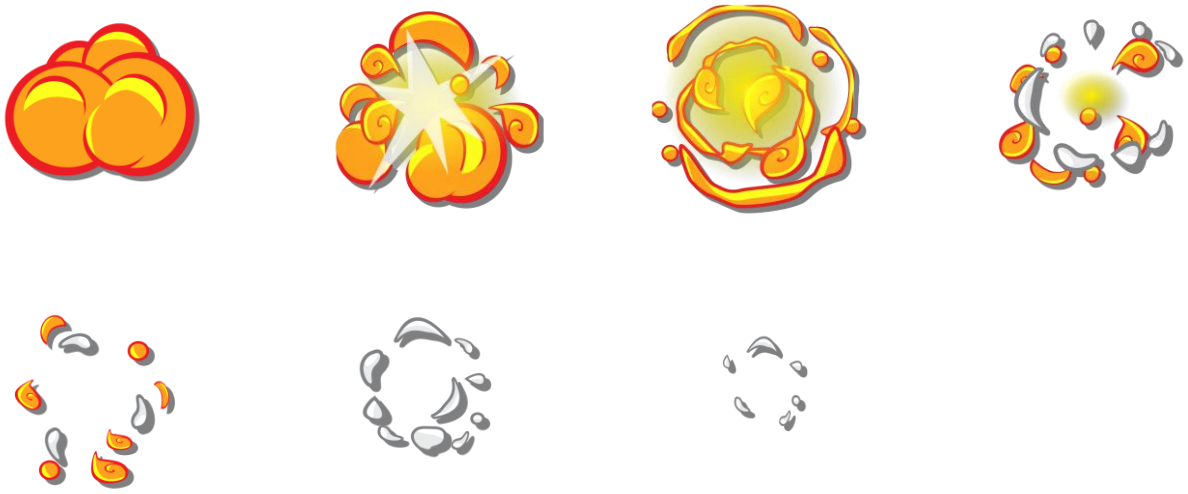
Animación que surge cuando pulsamos el botón mientras estamos tocando el suelo.

- **Volar:**

Animación que surge cuando estamos en un nivel vertical y podemos tener desde el principio del nivel un jet pack.

- **Explosión:**

Animación que surge cuando estamos en el nivel vertical y el jugador pierde la partida.

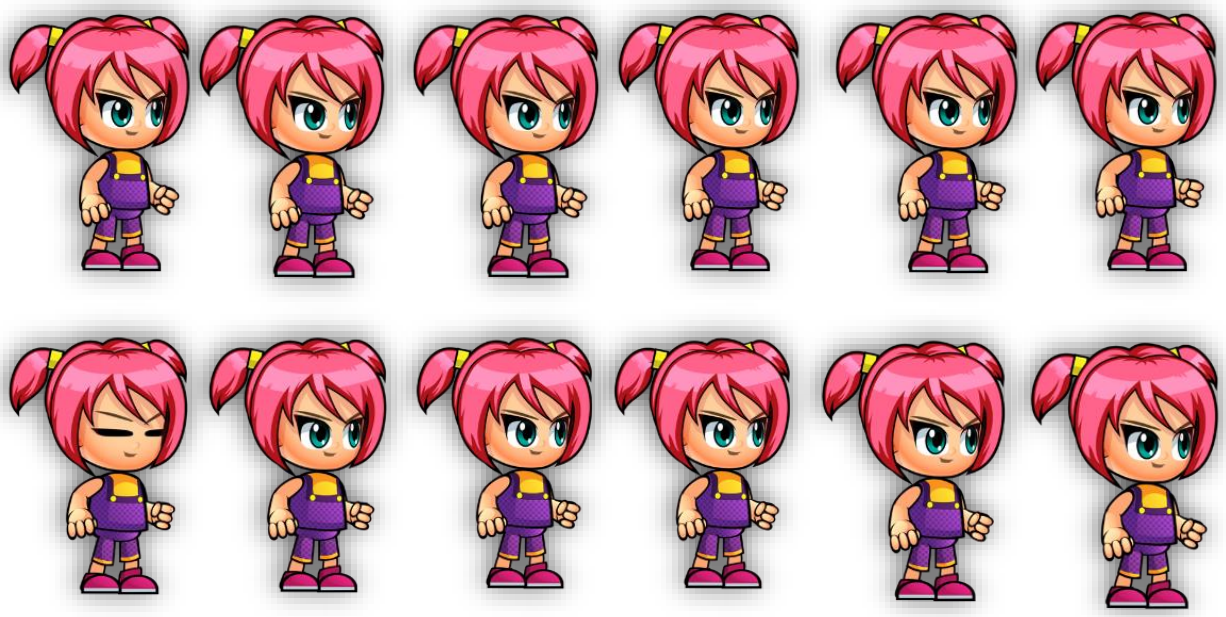


- **Herir:**

Animación que surge cuando algún enemigo le hace daño.

- **Parar/Idle:**

Animación que surge cuando el jugador no interactúa con nada.



- **Caminar:**

Animación que surge cuando movemos el personaje de izquierda a derecha utilizando el joystick.

- **Morir:**

Animación que surge cuando se nos acaban los corazones estando en un nivel horizontal.

VALENTINA

Es un personaje femenino que simplemente lo que hace es informar al jugador sobre el control del juego, es decir, es una manera de ayudar al jugador a conocer mejor el juego, como una especie de tutorial que el usuario puede decidir si hacerle caso o no. No le obliga a leer lo que tiene que decirle, si él quiere lo leerá e ira pasando las imágenes que mostrara el personaje Valentina, por tanto, es una decisión del usuario final.

ANIMACIONES

Cuando el personaje está dentro de la escena, estará esperando a detectar el jugador, en el momento que detecte al jugador, aparecerá una imagen como si estuviera hablándole, por tanto, tiene dos animaciones, que son:

- **Parar/Idle:**

Se produce en bucle mientras no detecta al jugador.

- **Hablar:**

Se produce en bucle en el momento que detecta al jugador cerca de ella en bucle hasta que deja de decir lo que tenga que decir y vuelve a la animación de antes.

REFERENCIAS

Este juego tiene algunas referencias a juegos que hemos visto con anterioridad, como, por ejemplo:

New Super Mario Bros

Referencias a las monedas, al entorno como si fueran dibujos, sin violencia…

Crash Bandicoot

Referencias a los Pick ups, al tema de destruir las cajas…

Shovel Knight

Referencias a los Checkpoints.

Metal Slug 3

Referencias a la mecánica de la protagonista a la hora de disparar cuando está alejada de las cajas y atacar con un sable cuando está cerca de ellas.

Hasta este punto llega el documento GDD (Game Design Document).

- **Código:**

```
using System.Collections;
using UnityEngine;
using CnControls;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class MovePlayer : MonoBehaviour {

    //  -------------VARIABLES OF PINK-------------
    float move, speedPlayer = 15f, forcePlayer = 10f; //variable for get movement, your speed and add force
    bool canShot = false, haveBullets = true, playerSword = false;

    //Animator Controller
    Animator animatorPlayer; //Variable for save Animator Controller

    //Physics and Position
    public Rigidbody2D player; //Variable for get physics
    public Collider2D colliderSword; //For simulation of sword
    Vector3 positionPlayer; //Variable where I save the position of Pink, for change when player takes a checkpoint

    //Change UI button
    public Image buttonGunSword; //Change UI button when player has a gun or sword
    public Sprite gun, sword;  //Change sprite in buttonGunSword

    //Sound
    AudioSource //All the sounds in the scene, these are saved in the array called soundsPlayer
        audioShot, audioJump, audioSword, audioButtons,
        audioCoin, audioDiamond, audioFinishLevel, audioMusic;
    public AudioSource[] soundsPlayer;

    //Variables for shoot (bullets)
    float speedBullet = 20f;
    public Rigidbody2D bullet;
    int numberBullets = 0;
    public Transform spawnBullet; //Position where bullets appear in the scene

    /*  -------------METHODS-------------
        Firts method called and only execution one time */
    void Start () {
        animatorPlayer = GetComponent<Animator>(); //Get component Animator Controller of Pink
        positionPlayer = transform.position; //Save position of Pink when begin level

        //Simulation of sword
        colliderSword.enabled = false;
        playerSword = false;

        //Initialize variables
        GameManager.playerDie = false; //Variable for know if Pink is dead
        GameManager.lifePlayer = 3; //Life of Pink
        GameManager.lifeArcher = 3; //Life of the Goddess enemy
        GameManager.lifetroll = 30; //Life of Troll
        GameManager.isPause = false; //Variable for know if Pink puts pause menu

        //Assign position in the array soundsPlayer for all sounds
        audioShot = soundsPlayer[0];
        audioJump = soundsPlayer[1];
        audioSword = soundsPlayer[2];
        audioButtons = soundsPlayer[3];
        audioCoin = soundsPlayer[4];
        audioDiamond = soundsPlayer[5];
        audioFinishLevel = soundsPlayer[6];
        audioMusic = soundsPlayer[7];

        //To check if it's on the music or not
        if (GameManager.music == true){
            audioMusic.Play();
        }

        //Variable for controll time in the game, when this is = 0, the time stop, when this is = 1, the time is normal
        Time.timeScale = 1;


        //Initialize variables of Level 1
        GameManager.level1Complete = false;
        GameManager.numberCoinsL1 = 0;
        GameManager.numberDiamondsL1 = 0;
        GameManager.numberEnemiesDestroyedL1 = 0;
    }
```

```
// Method is called once per frame
void Update () {

    //Method for that Player can move
    walk();

    /*  Method for that Player can shoot.
        You can shoot if touch button in the UI and you have less than 10 bullets fired and if player is live,
        3 conditions must be met, because the player has 10 bullets,
        when it comes to 10 have to wait a few seconds to reload your gun
    */
    if (canShot == true && numberBullets < 10 && GameManager.playerDie == false) //For know if the player looks to right or to left, this case is to right
    {
        if (GameManager.playerRight == true)
        {
            Rigidbody2D bulletInstance = Instantiate(bullet, spawnBullet.position, Quaternion.Euler(0, 0, 0)); //it creates the instance of the bullet
            bulletInstance.velocity = new Vector2(speedBullet, 0); //Apply the speed to the Bullet
            numberBullets++; //Counter of bullets
            if (numberBullets == 10)
            {
                GameManager.noBullets = true;
                StartCoroutine(loadBullets()); //Starts the coroutine for charge bullets
                haveBullets = false;
            }
            canShot = false;
        }
        else //it creates the instance of the bullet to left
        {
            Rigidbody2D bulletInstance = Instantiate(bullet, spawnBullet.position, Quaternion.Euler(0, 180, 0));
            bulletInstance.velocity = new Vector2(-speedBullet, 0);
            numberBullets++;
            if (numberBullets == 10)
            {
                GameManager.noBullets = true;
                StartCoroutine(loadBullets());
                haveBullets = false;
                }
                canShot = false;
            }
        }

        //When life of player is 0, player is dead
        if (GameManager.lifePlayer == 0)
         {
            GameManager.playerDie = true;
            animatorPlayer.SetBool("PlayerDie",true);
         }
    }

    //Couroutine for charge bullets
    IEnumerator loadBullets()
    {
        yield return new WaitForSeconds(3); //wait 3 seconds
        numberBullets = 0; //Now, player will can shoot
        GameManager.noBullets = false;
        haveBullets = true;
    }
 void walk()
 {
     move = CnInputManager.GetAxis("Horizontal"); //return value 1(right) or -1 (left)

     if (move > 0 && GameManager.playerDie == false)
     {
         animatorPlayer.SetBool("Move", true); //Start animation Move
         transform.rotation = Quaternion.Euler(0, 0, 0);
         transform.position += Vector3.right * speedPlayer * Time.deltaTime; //move to right the position Player
         GameManager.playerRight = true;
     }

     if (move < 0 && GameManager.playerDie == false)
     {
         animatorPlayer.SetBool("Move", true); //Start animation Move
         transform.rotation = Quaternion.Euler(0, 180, 0);
         transform.position += Vector3.left * speedPlayer * Time.deltaTime; //move to left the position Player
         GameManager.playerRight = false;
     }

     if (move == 0 && GameManager.playerDie == false) //When the player is calm
     {
         animatorPlayer.SetBool("Move", false); //Stop animation Move, therefore it will start animation Idle
     }
 }
```

```
 //When the player press the button of shoot
 public void shot()
 {
     // Mathf.Abs(player.velocity.y) < 0.01f It is to know if the player is touching the ground
     if (haveBullets == true && Mathf.Abs(player.velocity.y) < 0.01f && GameManager.playerDie == false && playerSword == false)
     {
         //To check if it's on the sound of shoot or not
         if (GameManager.sound == true) {
             audioShot.Play();
         }
         buttonGunSword.GetComponent<Image>().sprite = gun; //Change sprite button in UI
         animatorPlayer.SetTrigger("Shot"); //Start animation Shot
         canShot = true;
         playerSword = false;
     }

     //Player can attack with sword
     if (playerSword == true && GameManager.playerDie != true)
     {
         //To check if it's on the sound of sword or not
         if (GameManager.sound == true){
             audioSword.Play();
         }
         buttonGunSword.GetComponent<Image>().sprite = sword; //Change sprite button in UI
         animatorPlayer.SetTrigger("Sword"); //Start animation Sword
         colliderSword.enabled = true; //Appears the collider's sword
     }

 }

 //When the player stops touching the button
 public void stopShot()
 {
     canShot = false;
     colliderSword.enabled = false;

 }


//When the player press the button of jump
public void jump()
{
    //You can only jump if it touches the ground and player is alive
    if (Mathf.Abs(player.velocity.y)< 0.01f && GameManager.playerDie == false)
    {
        //To check if it's on the sound of shoot or not
        if (GameManager.sound == true){
            audioJump.Play();
        }
        animatorPlayer.SetTrigger("Jump"); //Start animation Jump
        player.AddForce(Vector3.up * forcePlayer, ForceMode2D.Impulse); //It applies to the player a force up
    }
    GameManager.playerPlaftormFinal = false;
}


 //When the player press the button of Pause
 public void pause()
 {
     //To check if it's on the sound of button or not
     if (GameManager.sound == true){
         audioButtons.Play();
     }
     GameManager.isPause = true;
     if (GameManager.isPause == true)
         Time.timeScale = 0; //Stop time
 }
```

```
//When the player press the button of Continue
public void ButtonContinue()
{
    //To check if it's on the sound of button or not
    if (GameManager.sound == true)
    {
        audioButtons.Play();
    }
    Time.timeScale = 1; //Time normal
    GameManager.isPause = false;

    //If player had finished level 1, when press button Continue he goes to levels menu
    if (GameManager.level1Complete == true)
    {
         SceneManager.LoadScene("LevelsMenu");
         GameManager.playerPlaftormFinal = false;
    }

    //If player had a checkpoint, reappear in the point of checkpoint
    if (GameManager.checkpoint == true )
    {
        GameManager.lifePlayer = 3;
        GameManager.playerDie = false;
        animatorPlayer.SetBool("PlayerDie",false);
        transform.position = positionPlayer;
        GameManager.playerPlaftormFinal = false;
    }
    else
    {
        GameManager.isPause = false;
        Time.timeScale = 1;
    }

    //If the player dies, it will charge the Level 1 from 0
    if (GameManager.playerDie == true)
    {
       SceneManager.LoadScene("Level1");
    }
}



//When the player press the button of Back
public void ButtonBackToLevelMenu()
{
    //To check if it's on the sound of button or not
    if (GameManager.sound == true){
        audioButtons.Play();
    }
    SceneManager.LoadScene("LevelsMenu");
    Time.timeScale = 1;
    GameManager.isPause = false;
}
```

```
//Method that is called when the player makes contact with a trigger
void OnTriggerEnter2D(Collider2D collider)
{
    //If player touchs a gameObject with tag is arm, these belong to the enemy trolls
    if (collider.gameObject.tag == "arm" && GameManager.playerDie != true)
    {
        animatorPlayer.SetTrigger("Hurt");
        GameManager.hurtPlayer = true;
        GameManager.lifePlayer--;
    }
    else
    {
        GameManager.hurtPlayer = false;
    }

    //If player touchs a gameObject with tag is coin
    if (collider.gameObject.tag == "coin" && GameManager.playerDie != true)
    {
        //To check if it's on the sound of coin or not
        if (GameManager.sound == true)
        {
            audioCoin.Play();
        }
        GameManager.numberCoinsL1++; //Counter of coins
        Destroy(collider.gameObject); //It destroys the coin when the player touches it
    }

    //If player touchs a gameObject with tag is diamond
    if (collider.gameObject.tag == "diamond" && GameManager.playerDie != true)
    {
        //To check if it's on the sound of diamond or not
        if (GameManager.sound == true){
            audioDiamond.Play();
        }
        GameManager.numberDiamondsL1++; //Counter of diamonds
        Destroy(collider.gameObject); //It destroys the diamond when the player touches it
    }

    //When the player drops to empty
    if (collider.gameObject.tag == "colliderFallDeath" && GameManager.playerDie != true)
    {
        GameManager.lifePlayer = 0;
    }

    //If player touchs a gameObject with tag is arrow, these arrows belong to the enemy of the goddesses
    if (collider.gameObject.tag == "arrow" && GameManager.playerDie != true)
    {
        animatorPlayer.SetTrigger("Hurt");
        GameManager.hurtPlayer = true;
        GameManager.lifePlayer -= 2;
    }
    else
    {
        GameManager.hurtPlayer = false;
    }
    //When the player touches a checkpoint, this changes the position of the player
    if (collider.gameObject.tag == "checkpoint" && GameManager.playerDie != true)
    {
        positionPlayer = transform.position;
    }

    //If player touchs a gameObject with tag is box
    if (collider.gameObject.tag == "box" && GameManager.playerDie != true)
    {
        playerSword = true;
    }
    else
    {
        playerSword = false;
    }
}
```

```
    //Method that is called when the player makes contact with a collider
    void OnCollisionEnter2D(Collision2D coll)
    {
        //If player touchs a spike in the level
        if (coll.gameObject.tag == "colliderDeath")
        {
            GameManager.lifePlayer = 0;
        }

        //When the player touches the final platform
        if (coll.gameObject.tag == "Finish")
        {
            GameManager.playerPlaftormFinal = true;
            if (GameManager.numberDiamondsL1 >= 3) //If player has 3 or more diamonds
            {
                //To check if it's on the sound of final level or not
                if (GameManager.sound == true)
                {
                    audioFinishLevel.Play();
                }
                GameManager.level1Complete = true;
            }
        }
    }
}
```

He escogido este fragmento de código que pertenece al script del control del personaje del nivel 1, es decir, el control del personaje de Pink, ya que me ha parecido lo más costoso y tedioso, porque tiene muchas animaciones, triggers, colliders… con las que puede interactuar, por eso lo más difícil ha sido encajarlo todo perfectamente.

En este proyecto, he creado todas las variables, métodos, comentarios e incluso los commits que he hecho para el repositorio, todo en inglés, porque es una buena manera de practicar el idioma aparte de que a la hora de mostrar a alguien lo que has hecho, el poder ver el código ordenado, comentado y realizado en inglés podrá leerlo mucha más gente y es más presencial para tu curriculum.

6. BIBLIOGRAFÍA

- https://www.assetstore.unity3d.com/en/
- https://unity3d.com/es
- https://www.freesound.org/
- http://www.piskelapp.com/
- https://trello.com/
- https://stackoverflow.com/
- https://github.com/
- https://www.youtube.com/
- https://graphicriver.net/
- https://play.google.com/store?hl=es